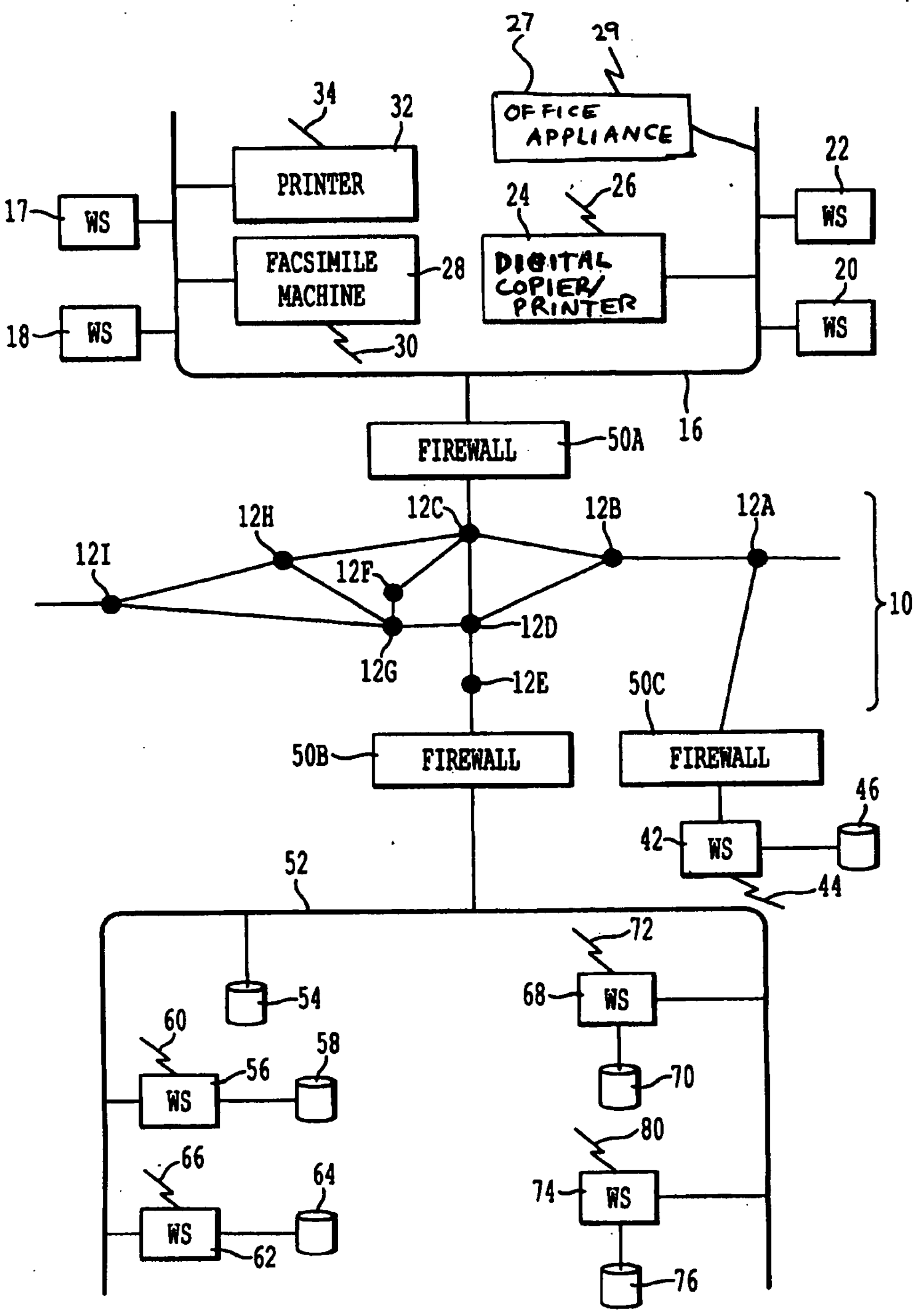

27
29
34
32
OFFICE APPLIANCE
PRINTER
22
17
WS
24
26
WS
FACSIMILE MACHINE
28
DIGITAL COPIER/ PRINTER
20
18
WS
WS
30
16
FIREWALL
50A
12C
12B
12A
12H
12I
12F
10
12D
12G
12E
50C
50B
FIREWALL
FIREWALL
46
42
WS
52
44
72
68
WS
54
60
58
56
70
WS
80
66
64
74
WS
WS
62
76

FIG. 1

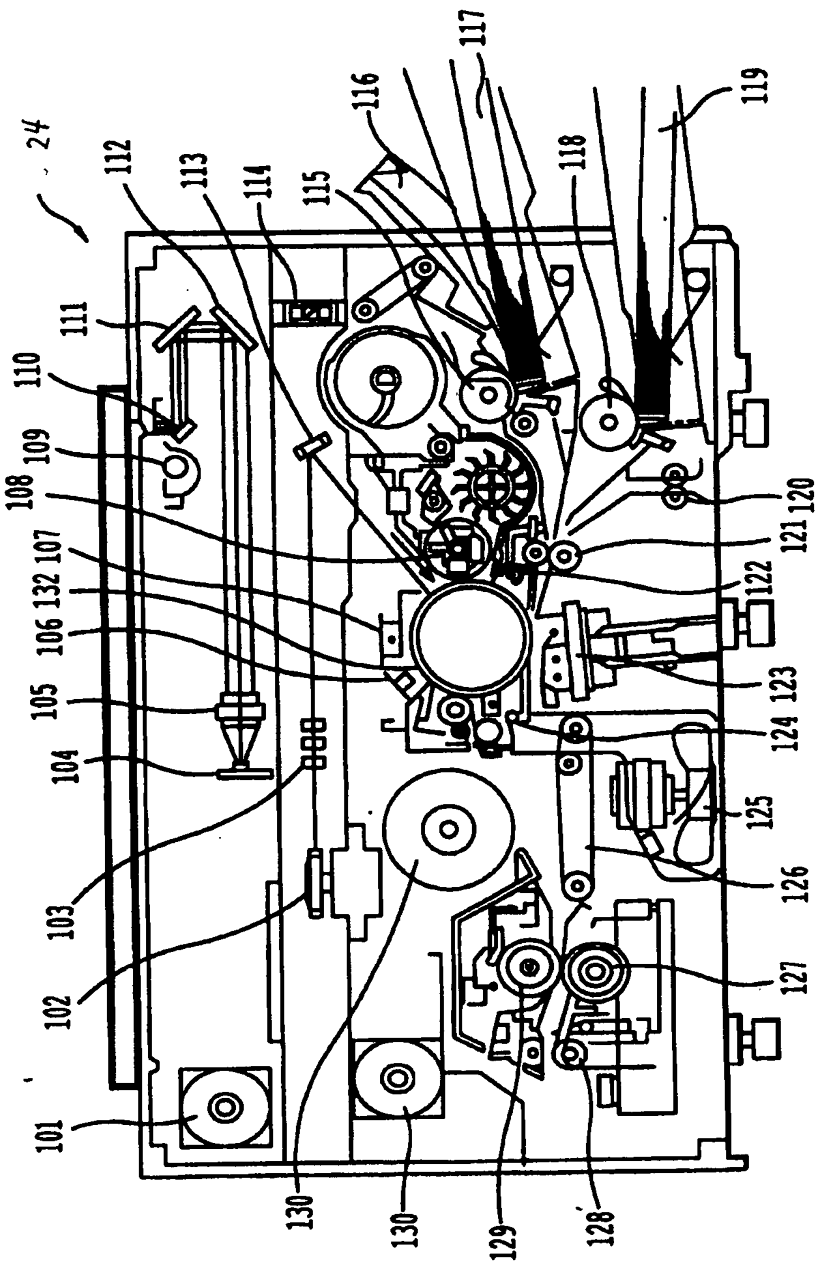

24
101
102
103
104
105
106
132
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
130

FIG.2

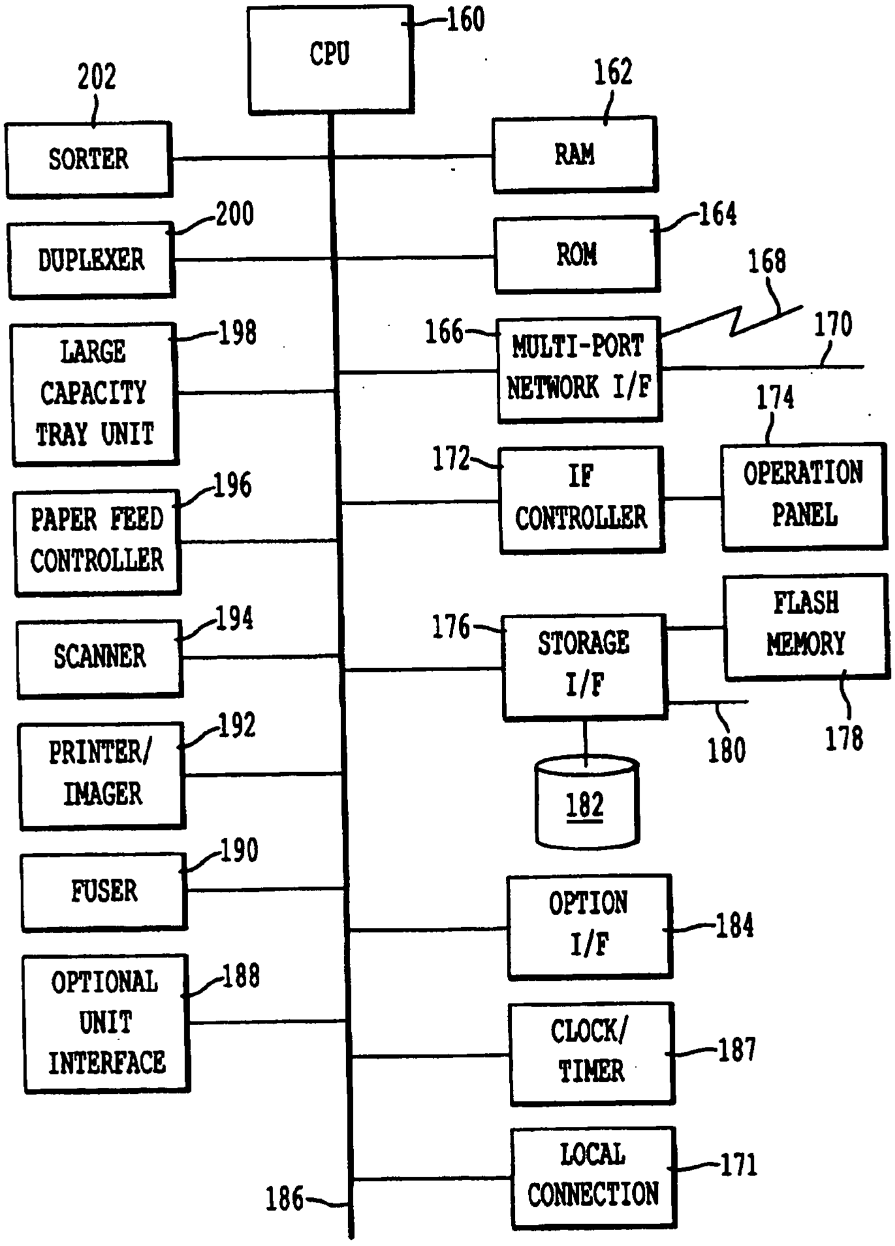

CPU
160
202
SORTER
162
RAM
200
DUPLEXER
164
ROM
168
198
LARGE CAPACITY TRAY UNIT
166
MULTI-PORT NETWORK I/F
170
174
196
PAPER FEED CONTROLLER
172
IF CONTROLLER
OPERATION PANEL
194
SCANNER
176
STORAGE I/F
FLASH MEMORY
180
178
192
PRINTER/ IMAGER
182
190
FUSER
OPTION I/F
184
188
OPTIONAL UNIT INTERFACE
CLOCK/ TIMER
187
LOCAL CONNECTION
171
186

FIG. 3

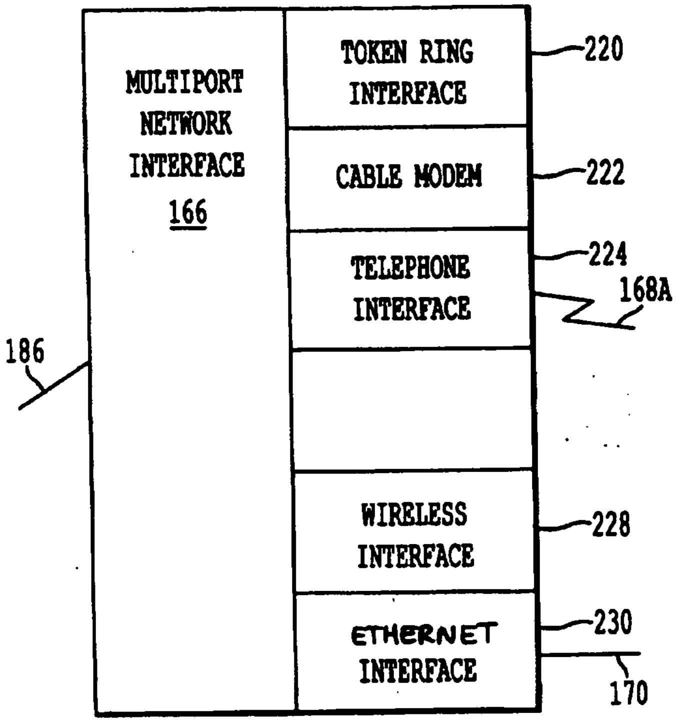

MULTIPORT NETWORK INTERFACE 166
TOKEN RING INTERFACE
220
CABLE MODEM
222
TELEPHONE INTERFACE
224
168A
186
WIRELESS INTERFACE
228
ETHERNET INTERFACE
230
170

FIG. 4

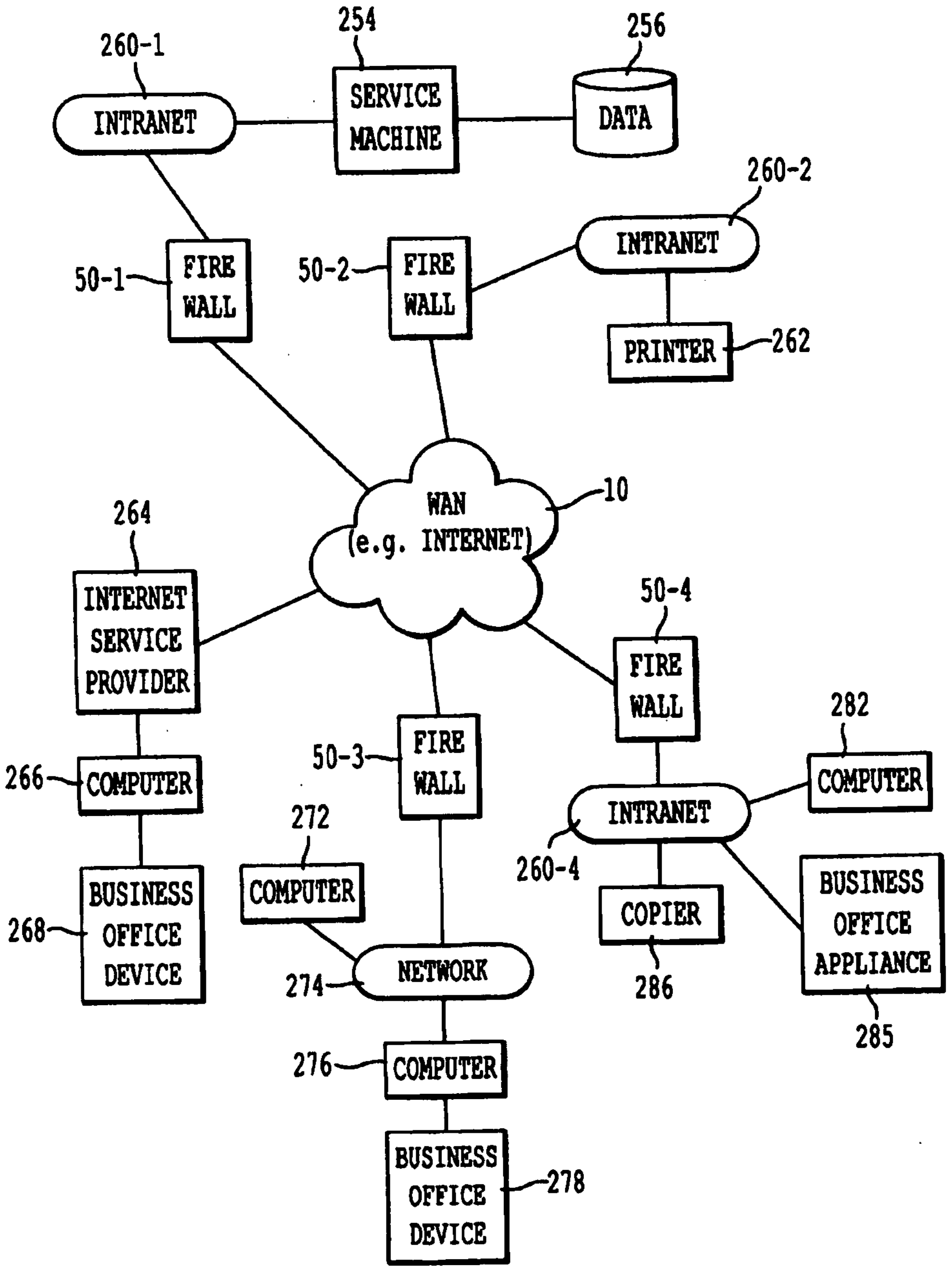

254
256
260-1
INTRANET
SERVICE MACHINE
DATA
260-2
INTRANET
50-1
FIRE WALL
50-2
FIRE WALL
PRINTER
262
WAN (e.g. INTERNET)
10
264
INTERNET SERVICE PROVIDER
50-4
FIRE WALL
282
266
COMPUTER
50-3
FIRE WALL
INTRANET
COMPUTER
272
260-4
BUSINESS OFFICE APPLIANCE
268
BUSINESS OFFICE DEVICE
COMPUTER
COPIER
274
NETWORK
286
285
276
COMPUTER
BUSINESS OFFICE DEVICE
278

FIG. 5

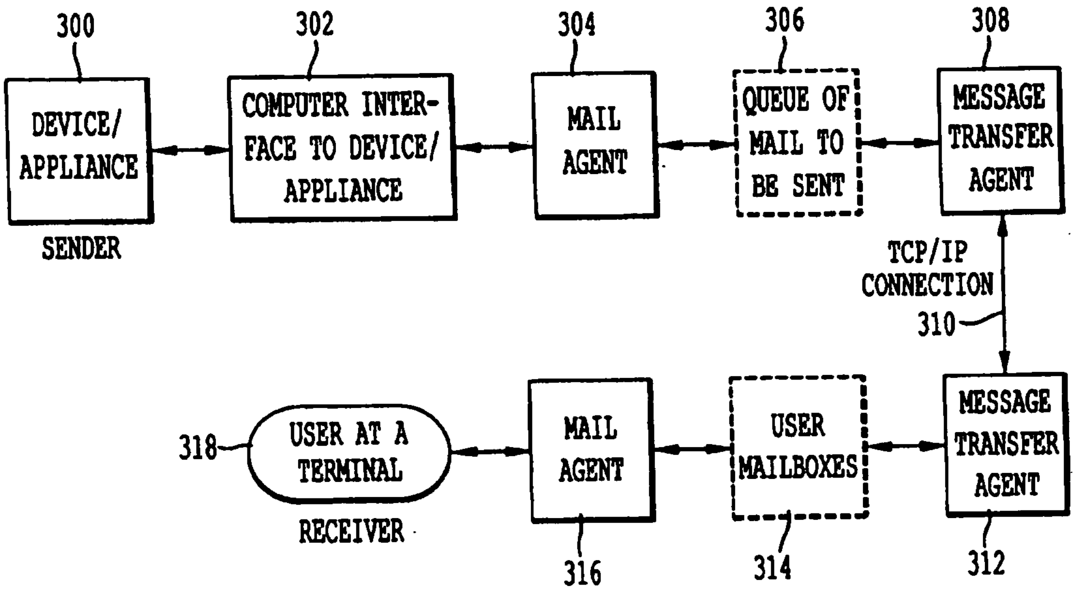

300
DEVICE/ APPLIANCE
SENDER
302
COMPUTER INTER-FACE TO DEVICE/ APPLIANCE
304
MAIL AGENT
306
QUEUE OF MAIL TO BE SENT
308
MESSAGE TRANSFER AGENT
TCP/IP CONNECTION
310
318
USER AT A TERMINAL
RECEIVER
316
MAIL AGENT
314
USER MAILBOXES
312
MESSAGE TRANSFER AGENT

FIG. 6A

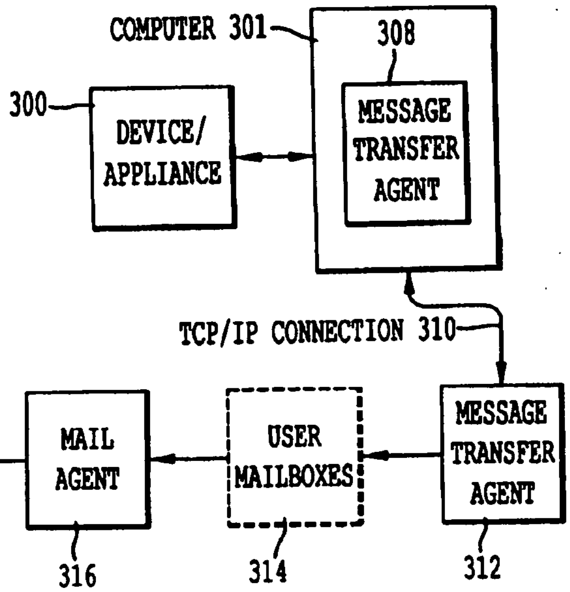

COMPUTER 301
308
MESSAGE TRANSFER AGENT
300
DEVICE/ APPLIANCE
TCP/IP CONNECTION 310
318
USER AT A TERMINAL
RECEIVER
316
MAIL AGENT
314
USER MAILBOXES
312
MESSAGE TRANSFER AGENT

FIG. 6B

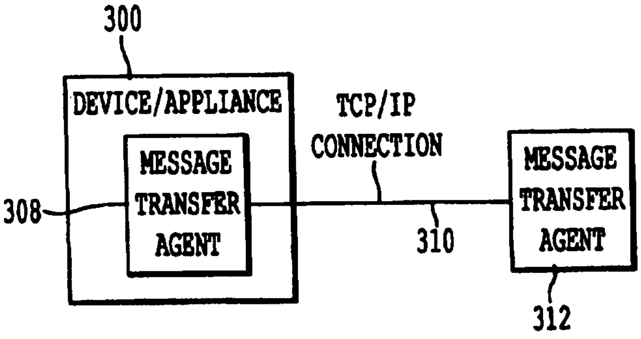

300
DEVICE/APPLIANCE
MESSAGE TRANSFER AGENT
308
TCP/IP CONNECTION
310
MESSAGE TRANSFER AGENT
312

FIG. 6C

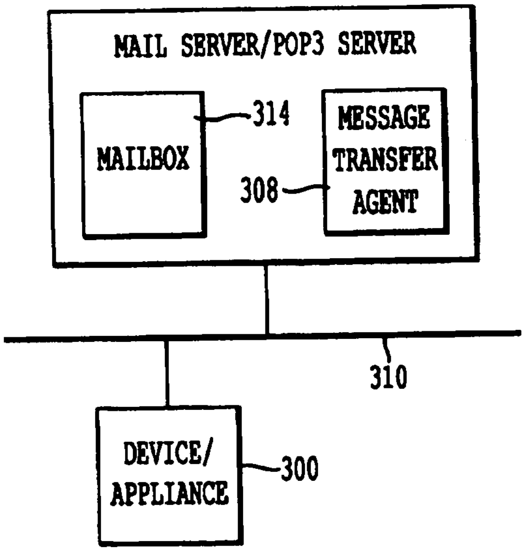

MAIL SERVER/POP3 SERVER
MAILBOX
314
MESSAGE TRANSFER AGENT
308
310
DEVICE/ APPLIANCE
300

FIG. 6D

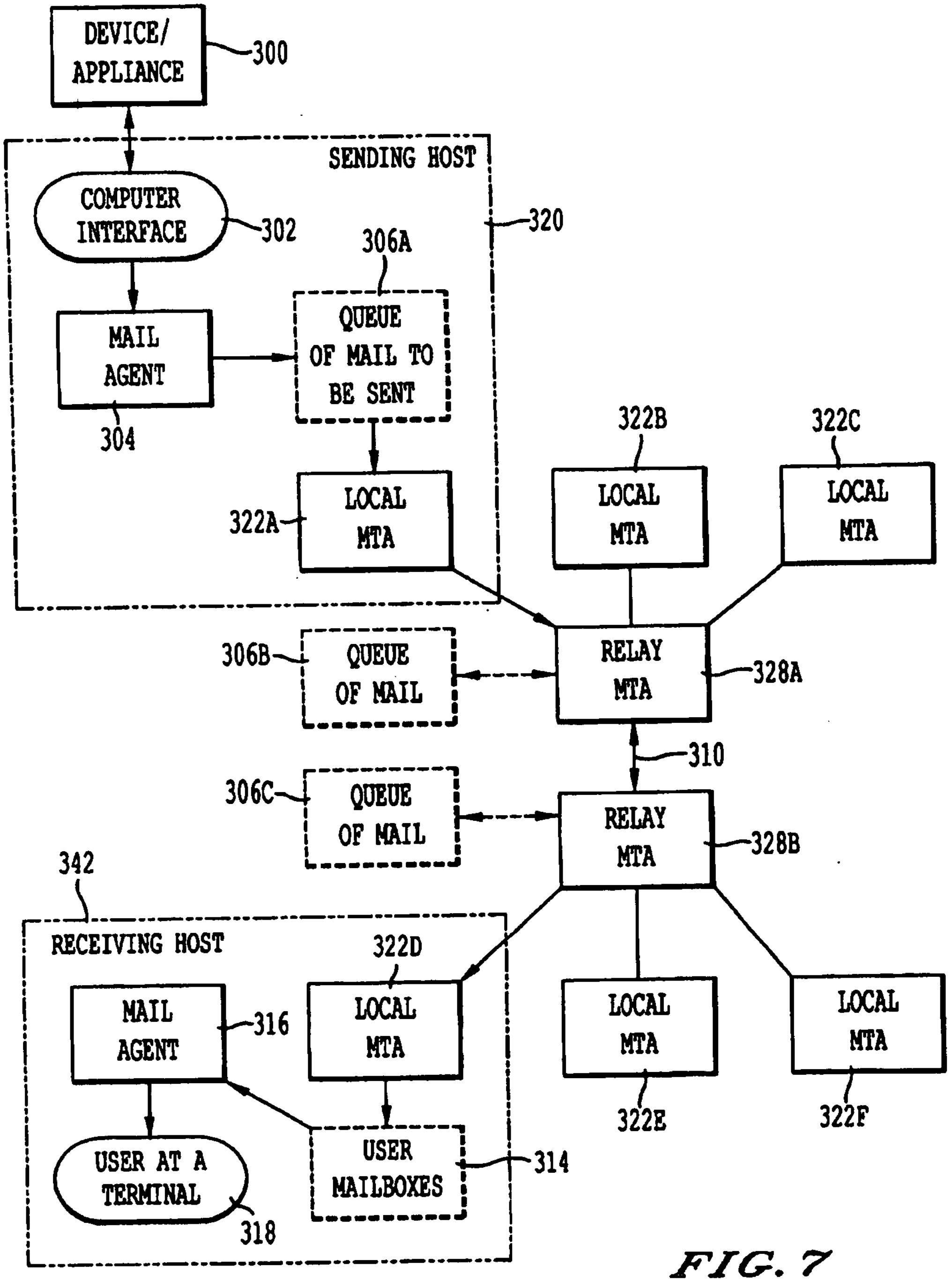

DEVICE/ APPLIANCE
300
SENDING HOST
COMPUTER INTERFACE
302
320
306A
MAIL AGENT
QUEUE OF MAIL TO BE SENT
304
322B
322C
LOCAL MTA
322A
LOCAL MTA
LOCAL MTA
306B
QUEUE OF MAIL
RELAY MTA
328A
310
306C
QUEUE OF MAIL
RELAY MTA
328B
342
RECEIVING HOST
322D
MAIL AGENT
316
LOCAL MTA
LOCAL MTA
LOCAL MTA
322E
322F
USER AT A TERMINAL
USER MAILBOXES
314
318

FIG. 7

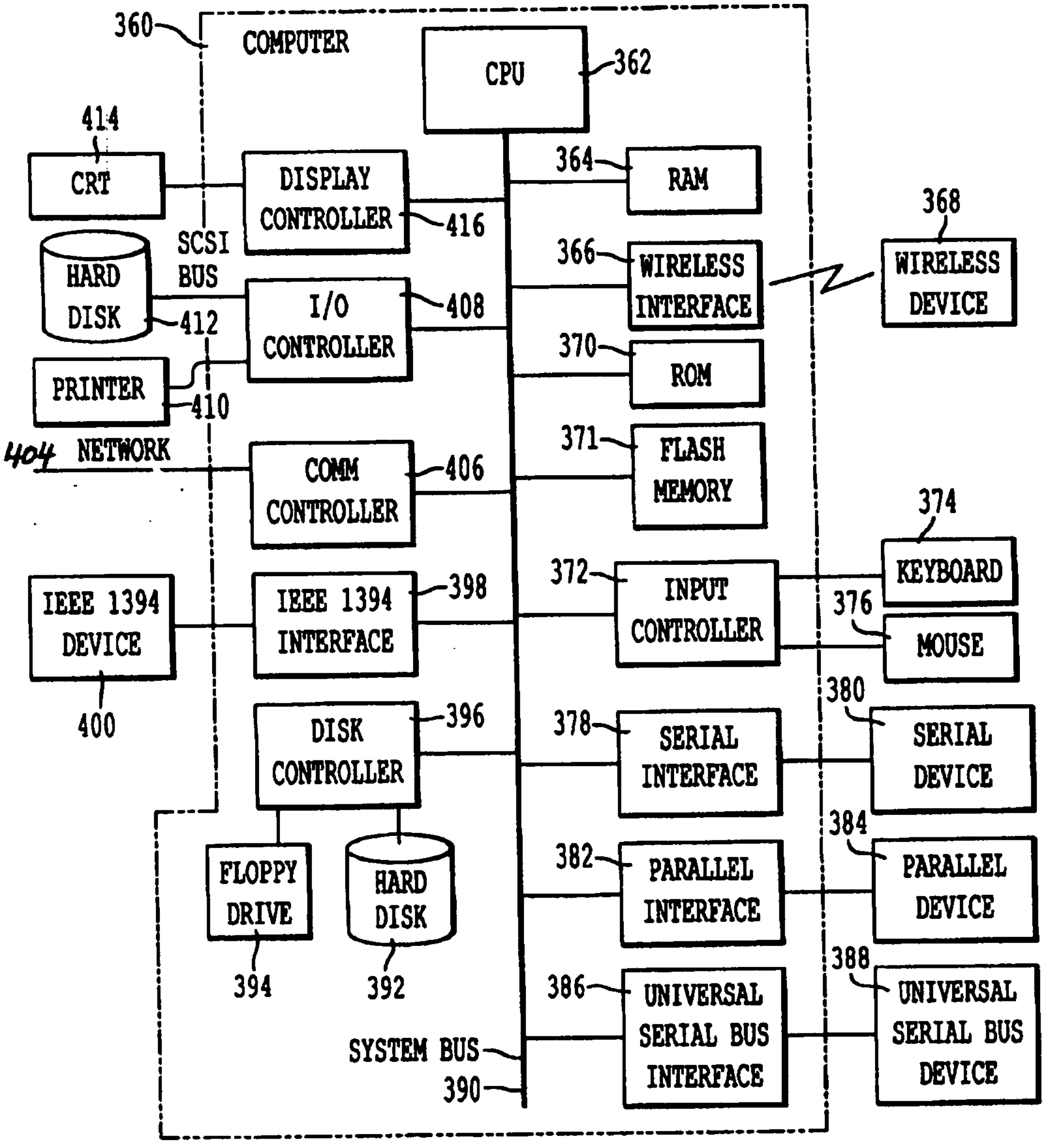

360
COMPUTER
CPU
362
414
CRT
DISPLAY CONTROLLER
416
364
RAM
SCSI BUS
HARD DISK
412
I/O CONTROLLER
408
366
WIRELESS INTERFACE
368
WIRELESS DEVICE
PRINTER
410
370
ROM
404 NETWORK
COMM CONTROLLER
406
371
FLASH MEMORY
374
KEYBOARD
IEEE 1394 DEVICE
400
IEEE 1394 INTERFACE
398
372
INPUT CONTROLLER
376
MOUSE
DISK CONTROLLER
396
378
SERIAL INTERFACE
380
SERIAL DEVICE
FLOPPY DRIVE
394
HARD DISK
392
382
PARALLEL INTERFACE
384
PARALLEL DEVICE
386
UNIVERSAL SERIAL BUS INTERFACE
388
UNIVERSAL SERIAL BUS DEVICE
SYSTEM BUS
390

FIG. 8

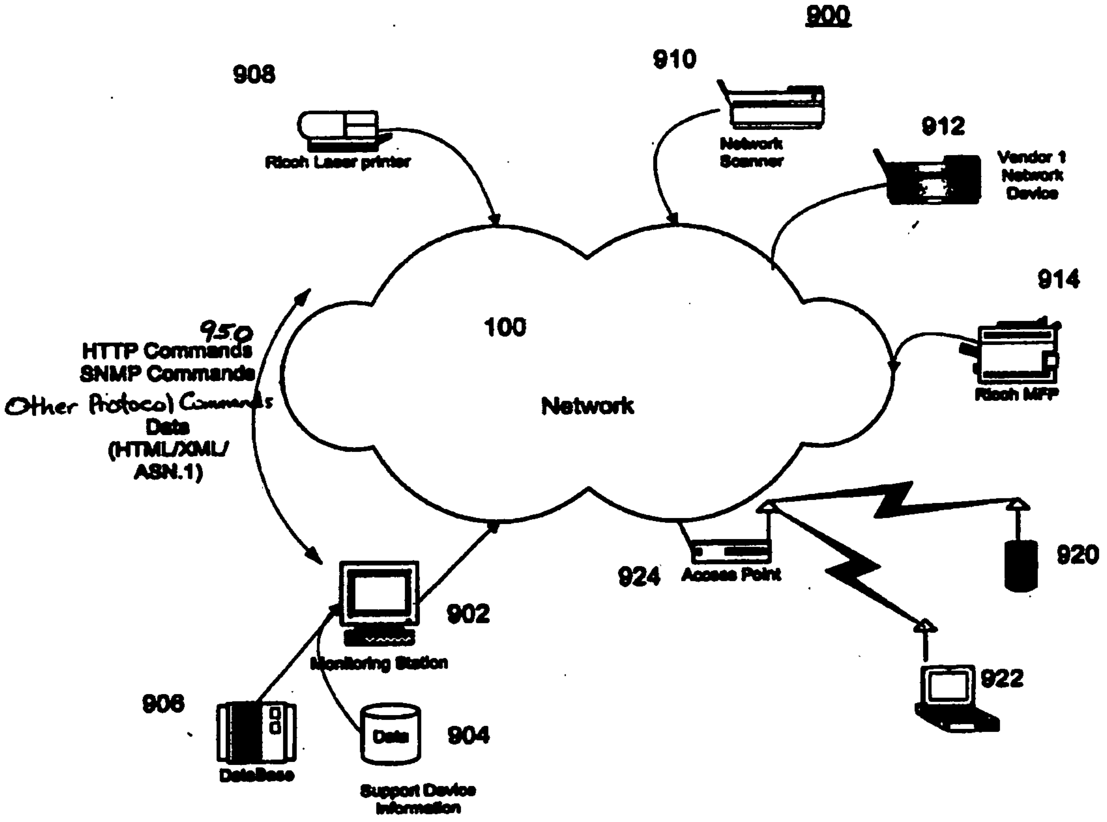

900
908
Ricoh Laser printer
910
Network Scanner
912
Vendor 1 Network Device
914
Ricoh MFP
100
Network
950
HTTP Commands
SNMP Commands
Other Protocol Commands
Data
(HTML/XML/
ASN.1)
920
924
Access Point
902
Monitoring Station
922
906
DataBase
Data
904
Support Device Information

Figure 9

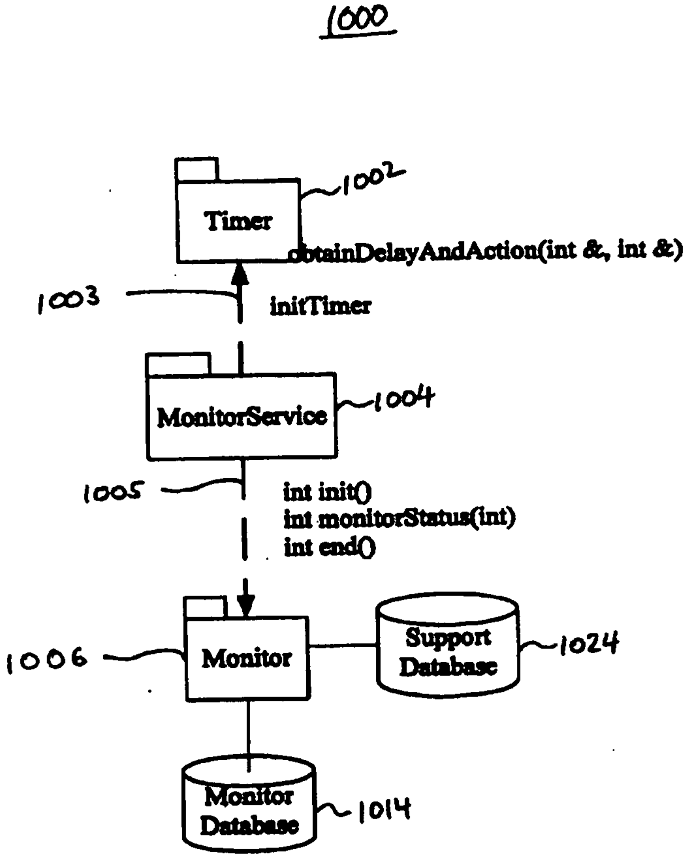
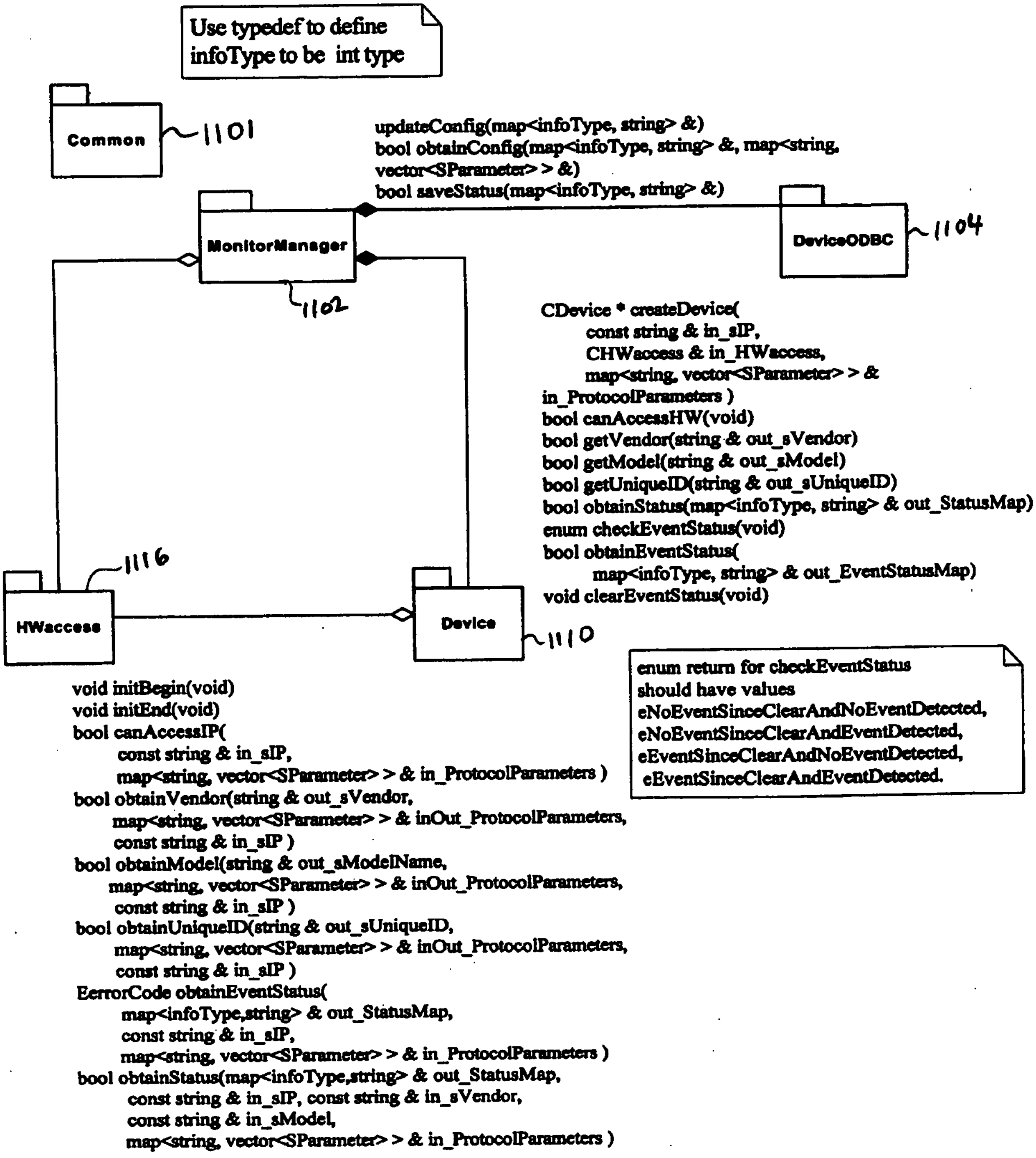

1000
Timer
1002
obtainDelayAndAction(int &, int &)
1003
initTimer
MonitorService
1004
1005
int init()
int monitorStatus(int)
int end()
1006
Monitor
Support Database
1024
Monitor Database
1014

FIG. 10



Use typedef to define infoType to be int type

```
updateConfig(map<infoType, string> &)
bool obtainConfig(map<infoType, string> &, map<string,
vector<SParameter> > &)
bool saveStatus(map<infoType, string> &)
```

```
CDevice * createDevice(
        const string & in_sIP,
        CHWaccess & in_HWaccess,
        map<string, vector<SParameter> > &
in_ProtocolParameters )
bool canAccessHW(void)
bool getVendor(string & out_sVendor)
bool getModel(string & out_sModel)
bool getUniqueID(string & out_sUniqueID)
bool obtainStatus(map<infoType, string> & out_StatusMap)
enum checkEventStatus(void)
bool obtainEventStatus(
        map<infoType, string> & out_EventStatusMap)
void clearEventStatus(void)
```

enum return for checkEventStatus should have values eNoEventSinceClearAndNoEventDetected, eNoEventSinceClearAndEventDetected, eEventSinceClearAndNoEventDetected, eEventSinceClearAndEventDetected.

```
void initBegin(void)
void initEnd(void)
bool canAccessIP(
        const string & in_sIP,
        map<string, vector<SParameter> > & in_ProtocolParameters )
bool obtainVendor(string & out_sVendor,
      map<string, vector<SParameter> > & inOut_ProtocolParameters,
      const string & in_sIP )
bool obtainModel(string & out_sModelName,
     map<string, vector<SParameter> > & inOut_ProtocolParameters,
      const string & in_sIP )
bool obtainUniqueID(string & out_sUniqueID,
      map<string, vector<SParameter> > & inOut_ProtocolParameters,
      const string & in_sIP )
EerrorCode obtainEventStatus(
       map<infoType,string> & out_StatusMap,
       const string & in_sIP,
       map<string, vector<SParameter> > & in_ProtocolParameters )
bool obtainStatus(map<infoType,string> & out_StatusMap,
        const string & in_sIP, const string & in_sVendor,
        const string & in_sModel,
        map<string, vector<SParameter> > & in_ProtocolParameters )
```

FIG. 11

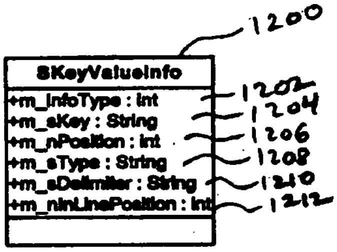

1200
SKeyValueInfo
+m_InfoType : int
+m_sKey : String
+m_nPosition : int
+m_sType : String
+m_sDelimiter : String
+m_nInLinePosition : int
1202
1204
1206
1208
1210
1212

FIG. 12

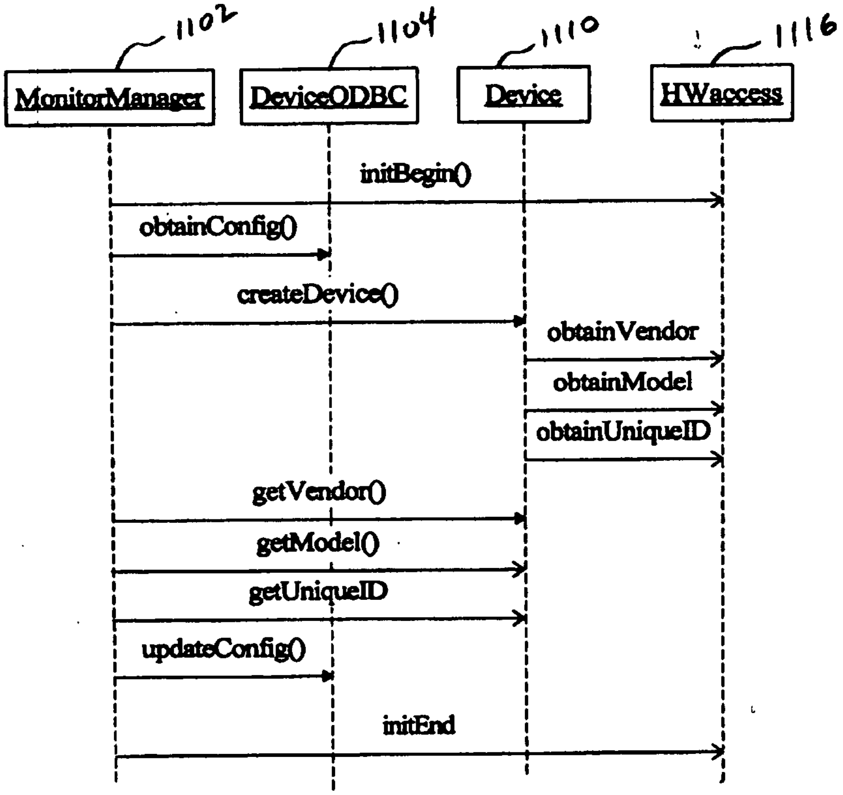

1102
1104
1110
1116
MonitorManager
DeviceODBC
Device
HWaccess
initBegin()
obtainConfig()
createDevice()
obtainVendor
obtainModel
obtainUniqueID
getVendor()
getModel()
getUniqueID
updateConfig()
initEnd

FIG. 13

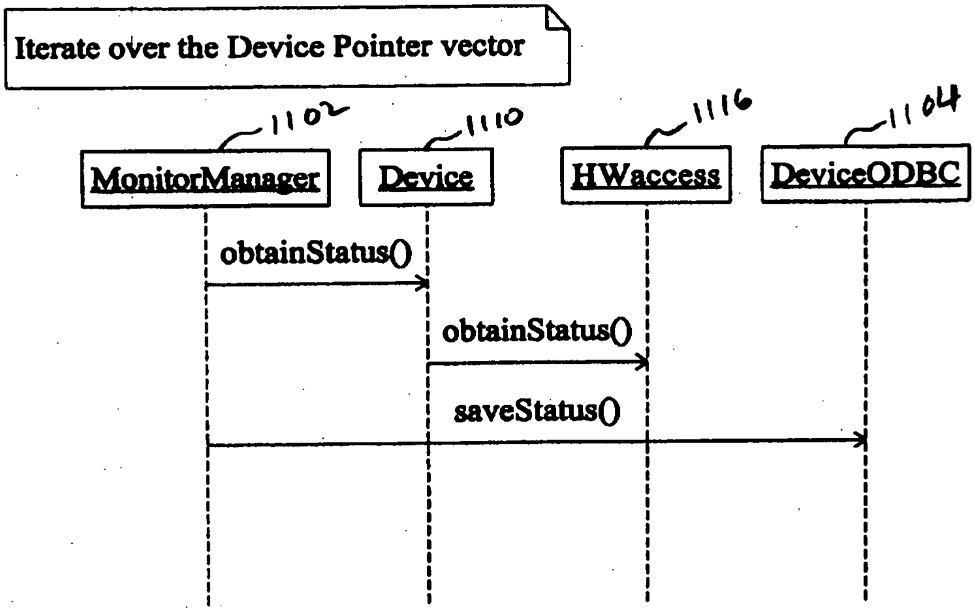
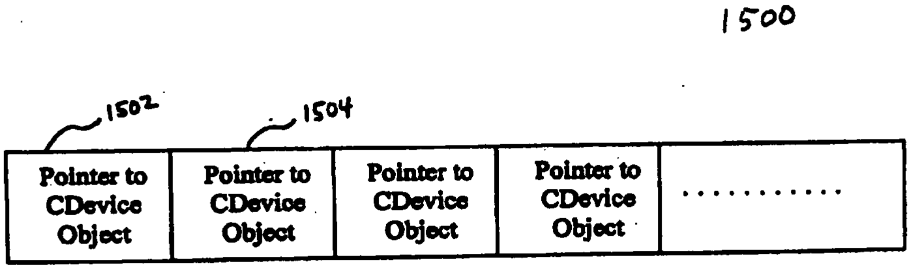

Iterate over the Device Pointer vector
1102
1110
1116
1104
MonitorManager
Device
HWaccess
DeviceODBC
obtainStatus()
obtainStatus()
saveStatus()

FIG. 14


1500
1502
1504
Pointer to CDevice Object
Pointer to CDevice Object
Pointer to CDevice Object
Pointer to CDevice Object
....

std::vector <pointer of CDevice >

FIG. 15

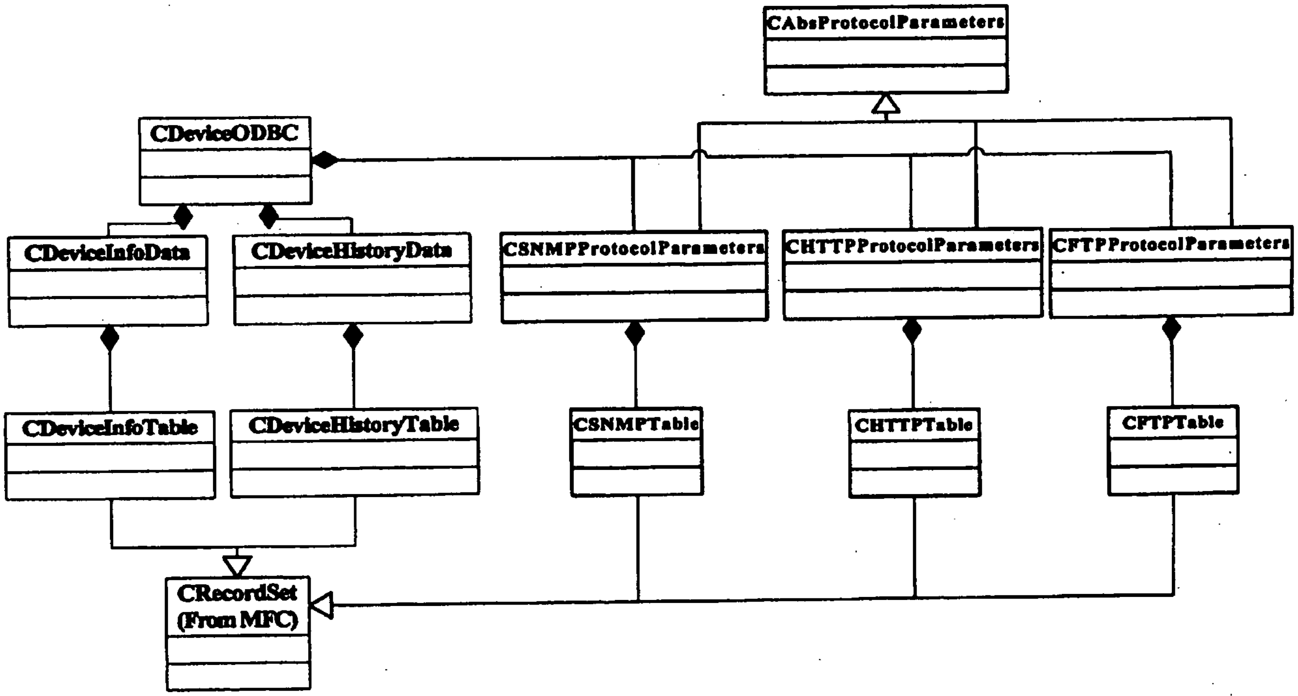

CAbsProtocolParameters
CDeviceODBC
CDeviceInfoData
CDeviceHistoryData
CSNMPProtocolParameters
CHTTPProtocolParameters
CFTPProtocolParameters
CDeviceInfoTable
CDeviceHistoryTable
CSNMPTable
CHTTPTable
CFTPTable
CRecordSet
(From MFC)

FIG. 16

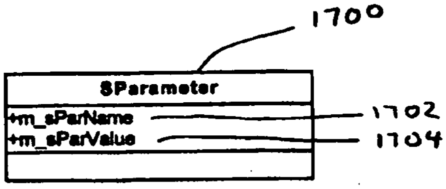
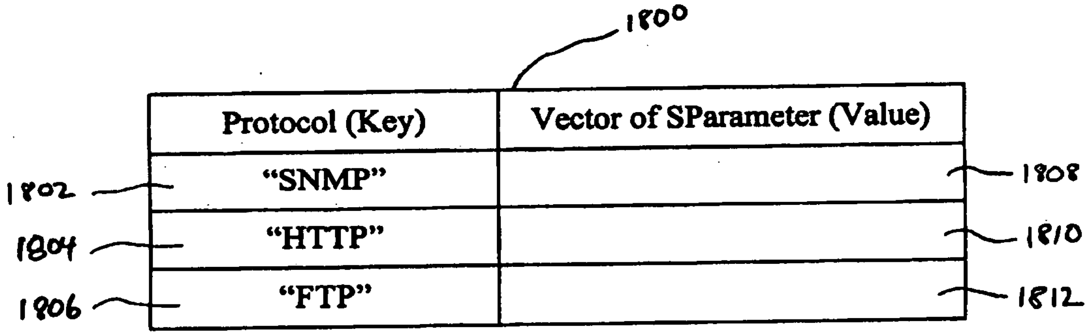


FIG. 17


1800
1802
1804
1806
1808
1810
1812

Protocol (Key)	Vector of SParameter (Value)
"SNMP"	
"HTTP"	
"FTP"	

FIG. 18

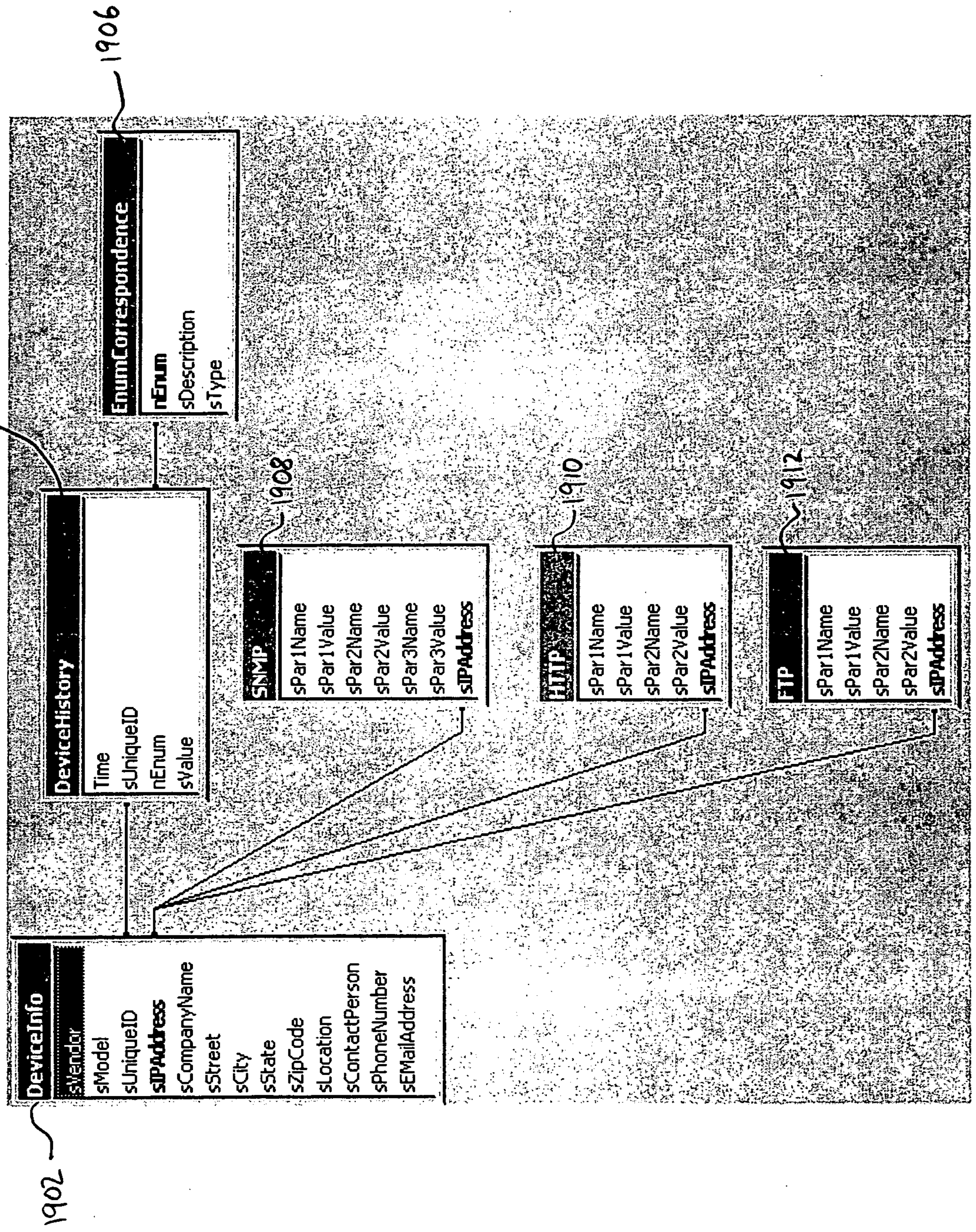

1902
DeviceInfo
sVendor
sModel
sUniqueID
sIPAddress
sCompanyName
sStreet
sCity
sState
sZipCode
sLocation
sContactPerson
sPhoneNumber
sEMailAddress
1904
DeviceHistory
Time
sUniqueID
nEnum
sValue
1906
EnumCorrespondence
nEnum
sDescription
sType
1908
SNMP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sPar3Name
sPar3Value
sIPAddress
1910
HTTP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sIPAddress
1912
FTP
sPar1Name
sPar1Value
sPar2Name
sPar2Value
sIPAddress

FIG. 19

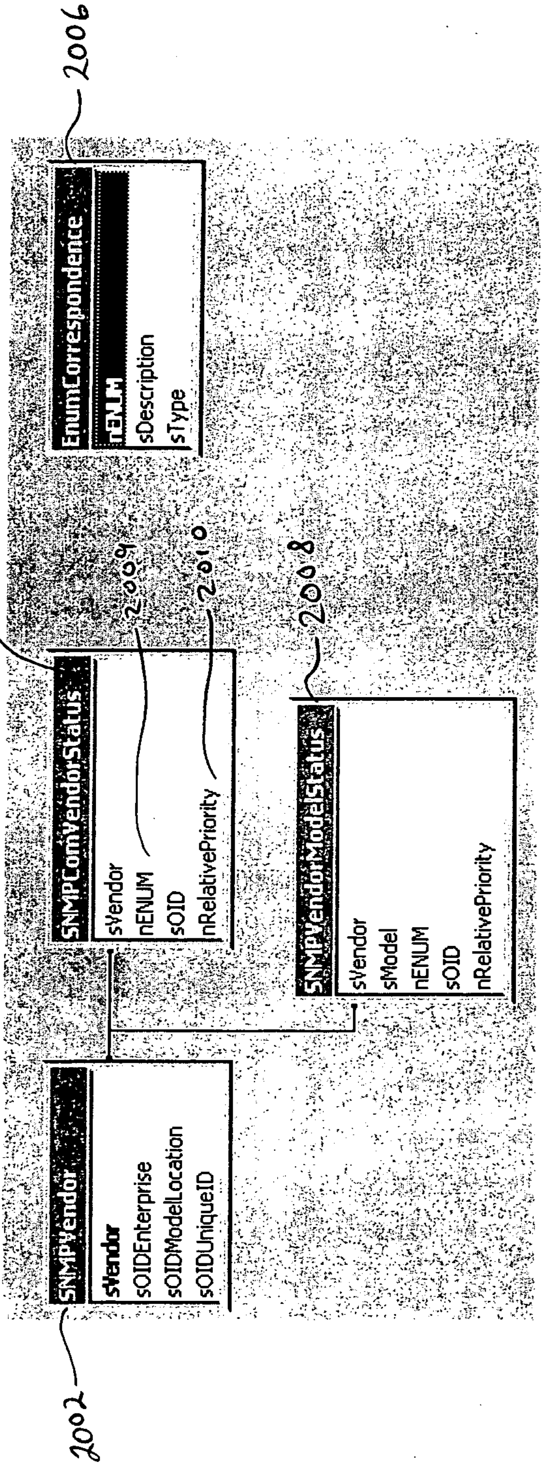
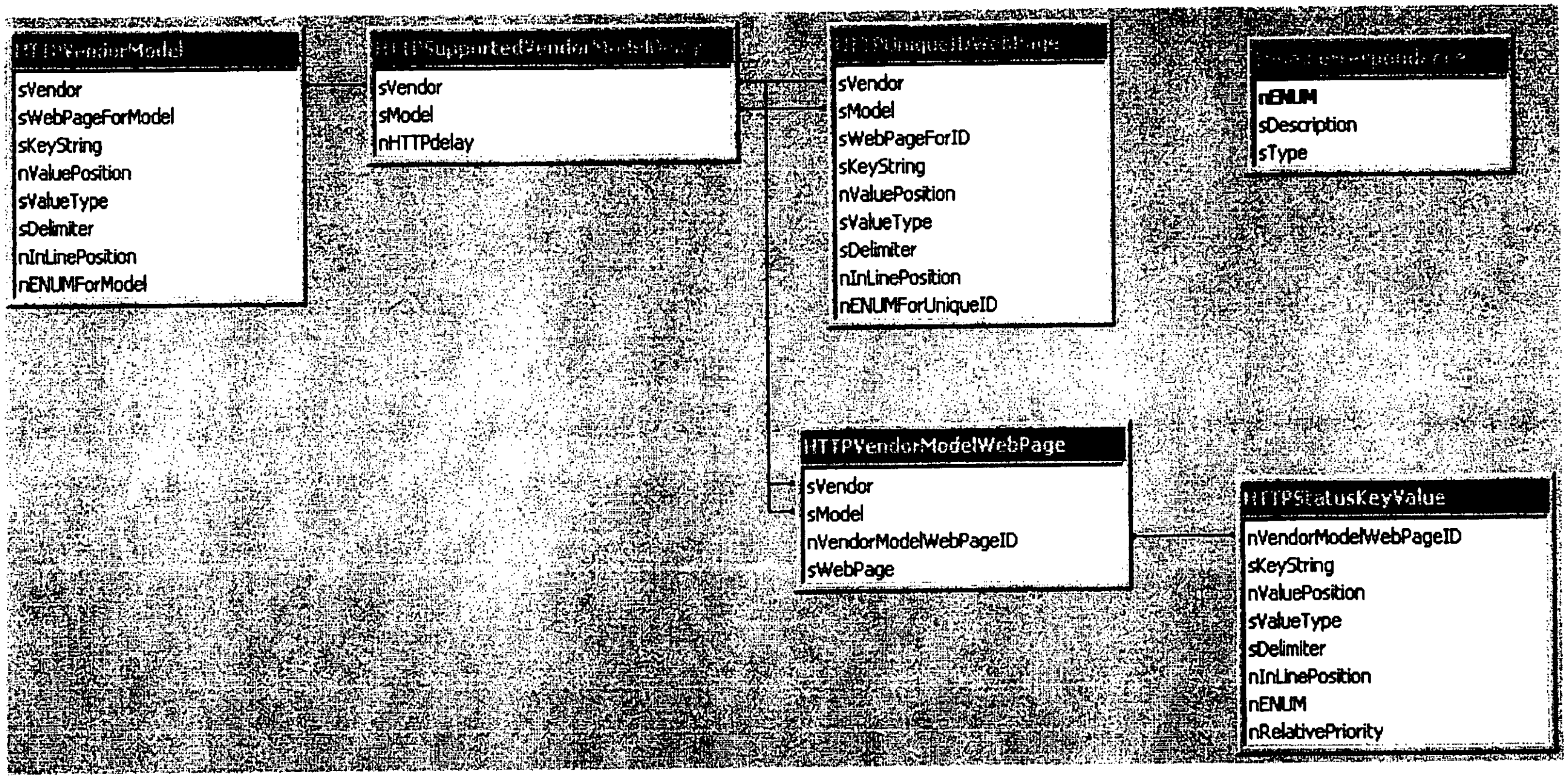

2002
SNMPVendor
sVendor
sOIDEnterprise
sOIDModelLocation
sOIDUniqueID
2004
SNMPComVendorStatus
sVendor
nENUM
sOID
nRelativePriority
2009
2010
2008
SNMPVendorModelStatus
sVendor
sModel
nENUM
sOID
nRelativePriority
2006
EnumCorrespondence
nENUM
sDescription
sType

FIG. 20



HTTPVendorModel
- sVendor
- sWebPageForModel
- sKeyString
- nValuePosition
- sValueType
- sDelimiter
- nInLinePosition
- nENUMForModel

HTTPSupportedVendorModelDelay
- sVendor
- sModel
- nHTTPdelay

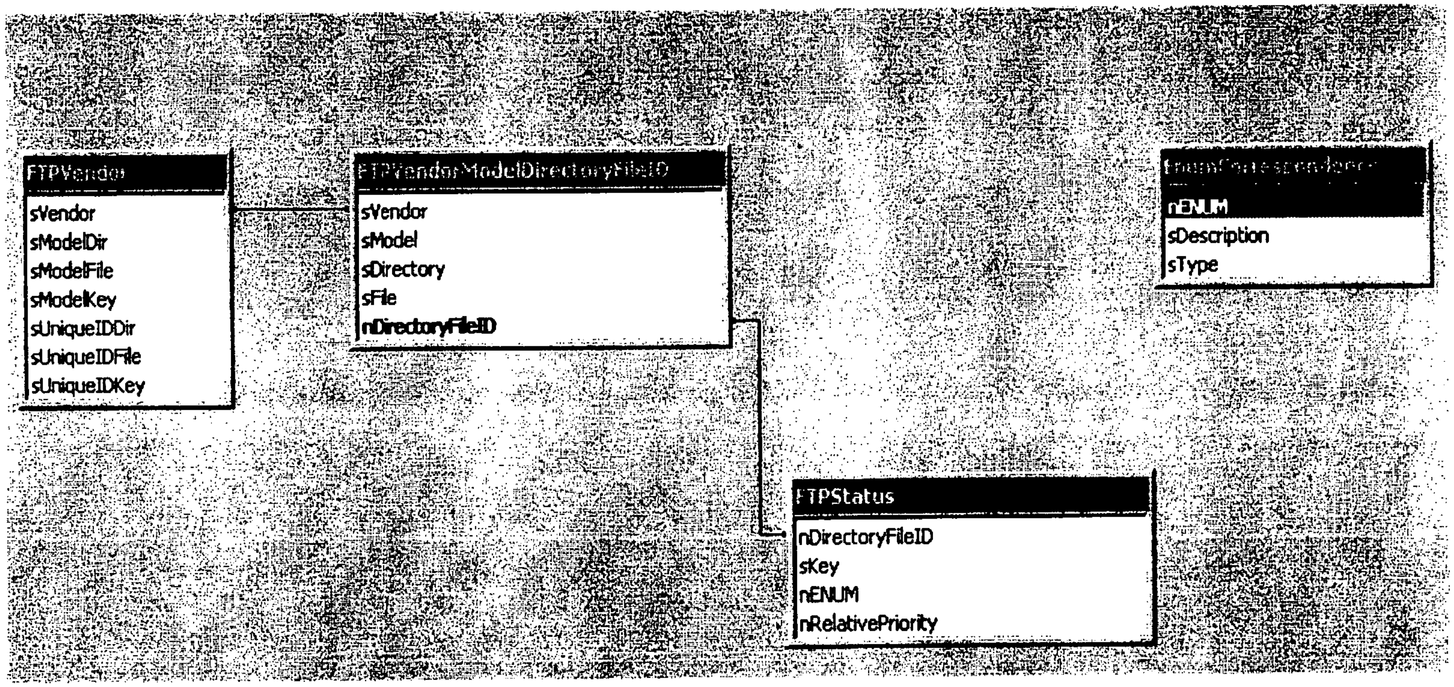
HTTPUniqueIDWebPage
- sVendor
- sModel
- sWebPageForID
- sKeyString
- nValuePosition
- sValueType
- sDelimiter
- nInLinePosition
- nENUMForUniqueID

[illegible]
- nENUM
- sDescription
- sType

HTTPVendorModelWebPage
- sVendor
- sModel
- nVendorModelWebPageID
- sWebPage

HTTPStatusKeyValue
- nVendorModelWebPageID
- sKeyString
- nValuePosition
- sValueType
- sDelimiter
- nInLinePosition
- nENUM
- nRelativePriority

FIG. 21



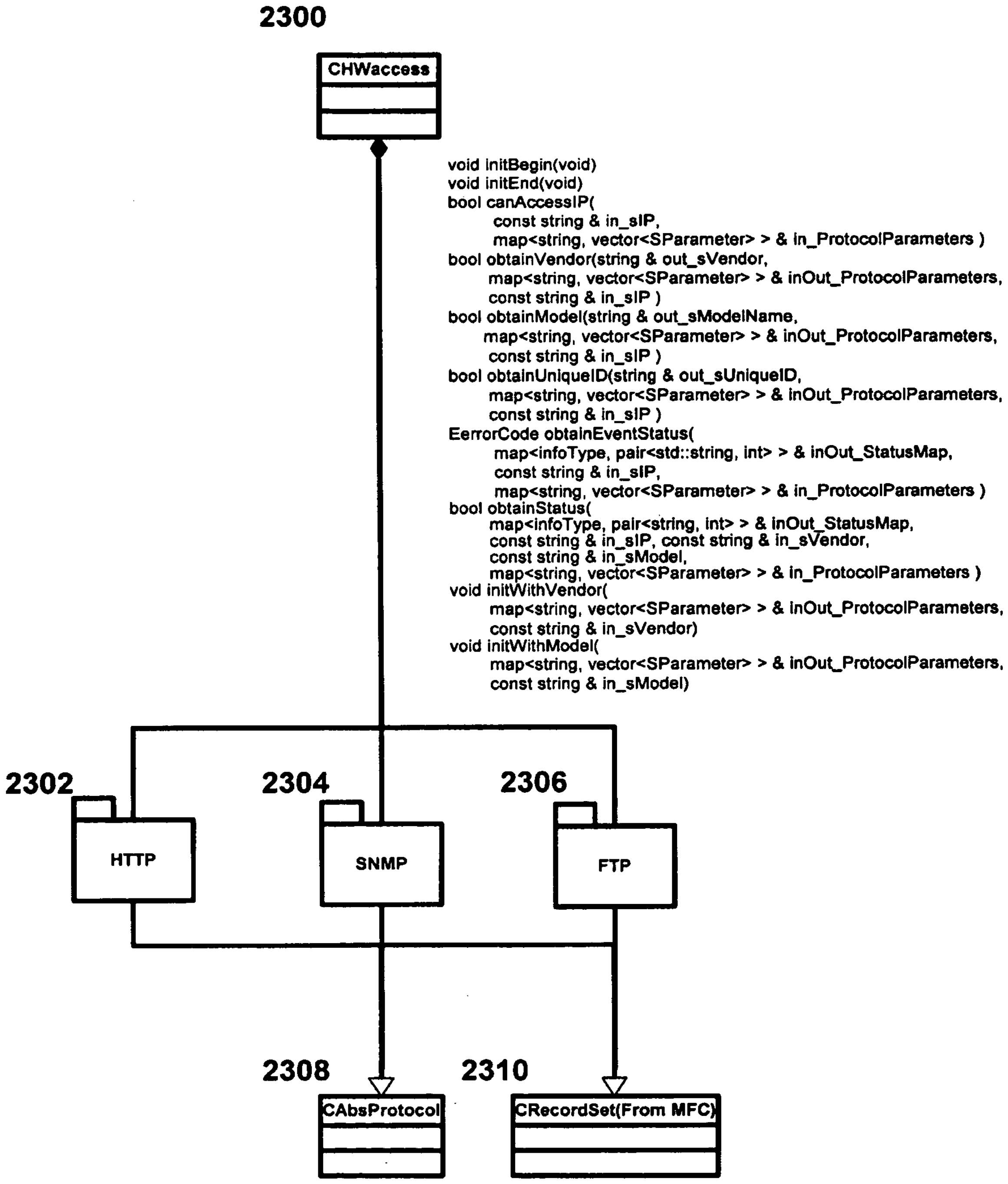
FTPVendor
sVendor
sModelDir
sModelFile
sModelKey
sUniqueIDDir
sUniqueIDFile
sUniqueIDKey

FTPVendorModelDirectoryFileID
sVendor
sModel
sDirectory
sFile
nDirectoryFileID

EnumCorrespondence
nENUM
sDescription
sType

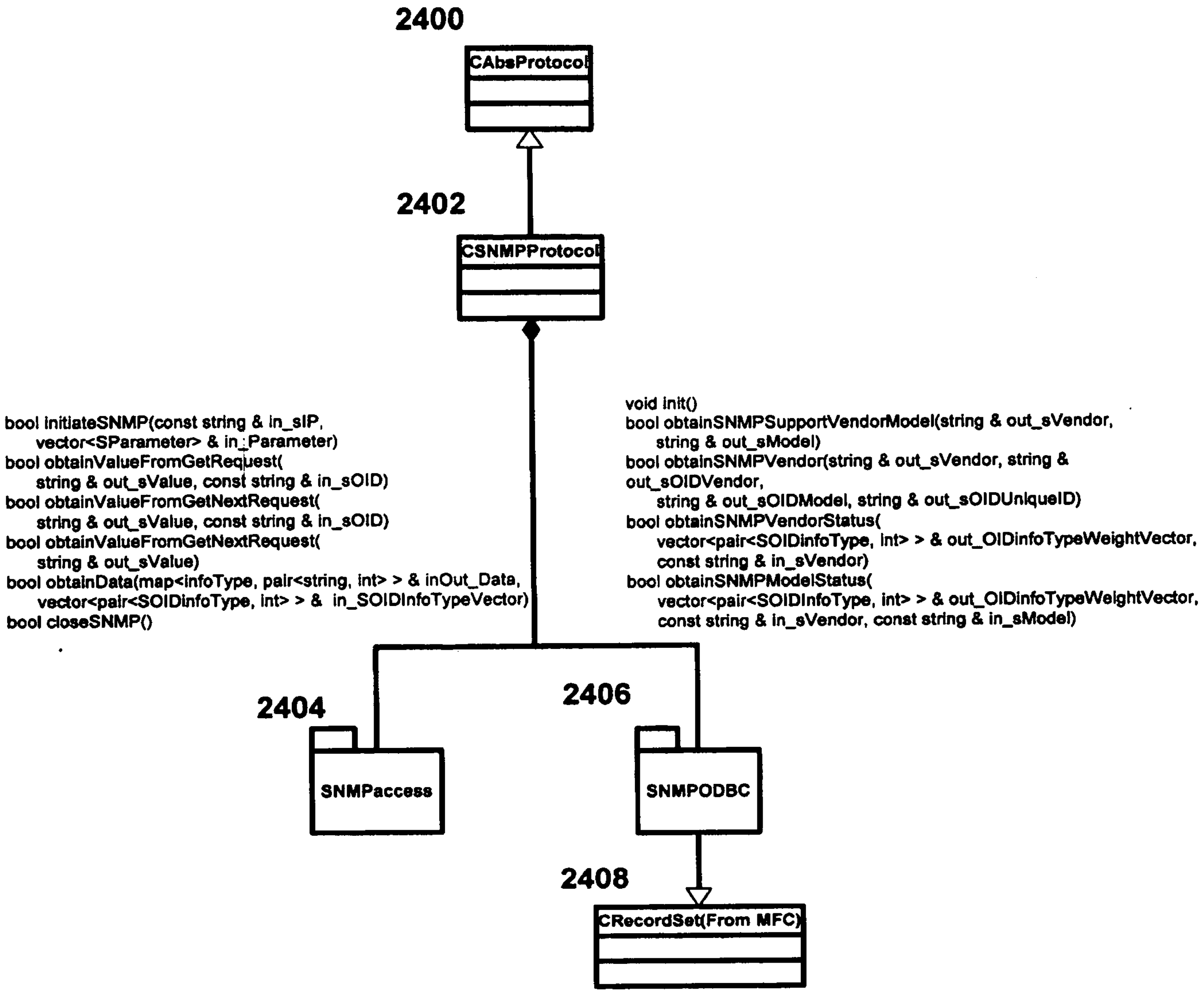
FTPStatus
nDirectoryFileID
sKey
nENUM
nRelativePriority

FIG. 22



```
void initBegin(void)
void initEnd(void)
bool canAccessIP(
        const string & in_sIP,
        map<string, vector<SParameter> > & in_ProtocolParameters )
bool obtainVendor(string & out_sVendor,
       map<string, vector<SParameter> > & inOut_ProtocolParameters,
       const string & in_sIP )
bool obtainModel(string & out_sModelName,
      map<string, vector<SParameter> > & inOut_ProtocolParameters,
       const string & in_sIP )
bool obtainUniqueID(string & out_sUniqueID,
       map<string, vector<SParameter> > & inOut_ProtocolParameters,
       const string & in_sIP )
EerrorCode obtainEventStatus(
        map<infoType, pair<std::string, int> > & inOut_StatusMap,
        const string & in_sIP,
        map<string, vector<SParameter> > & in_ProtocolParameters )
bool obtainStatus(
       map<infoType, pair<string, int> > & inOut_StatusMap,
       const string & in_sIP, const string & in_sVendor,
       const string & in_sModel,
       map<string, vector<SParameter> > & in_ProtocolParameters )
void initWithVendor(
       map<string, vector<SParameter> > & inOut_ProtocolParameters,
       const string & in_sVendor)
void initWithModel(
       map<string, vector<SParameter> > & inOut_ProtocolParameters,
       const string & in_sModel)
```

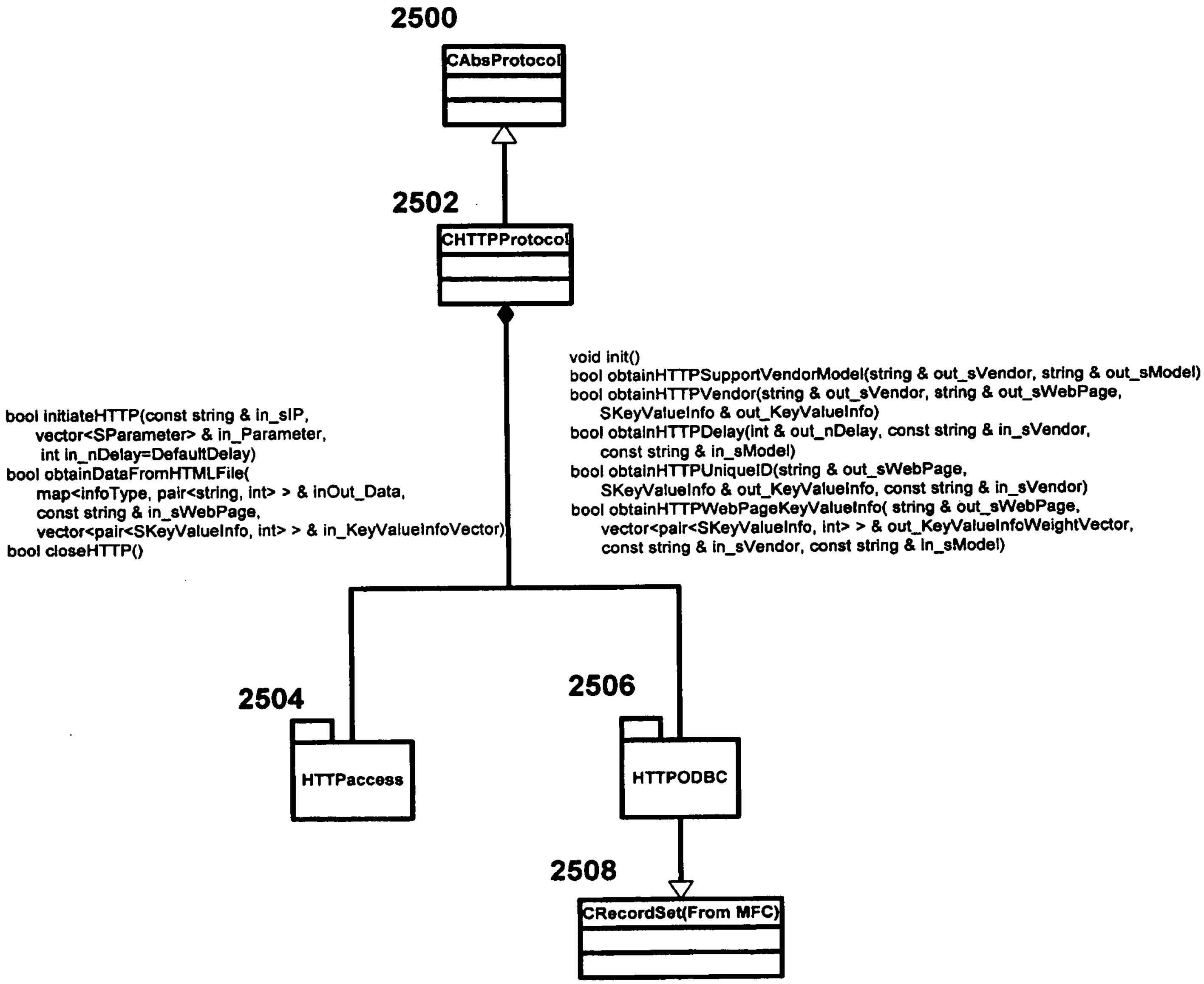
FIG. 23



```
bool initiateSNMP(const string & in_sIP,
    vector<SParameter> & in_Parameter)
bool obtainValueFromGetRequest(
    string & out_sValue, const string & in_sOID)
bool obtainValueFromGetNextRequest(
    string & out_sValue, const string & in_sOID)
bool obtainValueFromGetNextRequest(
    string & out_sValue)
bool obtainData(map<infoType, pair<string, int> > & inOut_Data,
    vector<pair<SOIDinfoType, int> > &  in_SOIDInfoTypeVector)
bool closeSNMP()
```

```
void init()
bool obtainSNMPSupportVendorModel(string & out_sVendor,
    string & out_sModel)
bool obtainSNMPVendor(string & out_sVendor, string &
out_sOIDVendor,
    string & out_sOIDModel, string & out_sOIDUniqueID)
bool obtainSNMPVendorStatus(
    vector<pair<SOIDinfoType, int> > & out_OIDinfoTypeWeightVector,
    const string & in_sVendor)
bool obtainSNMPModelStatus(
    vector<pair<SOIDInfoType, int> > & out_OIDinfoTypeWeightVector,
    const string & in_sVendor, const string & in_sModel)
```

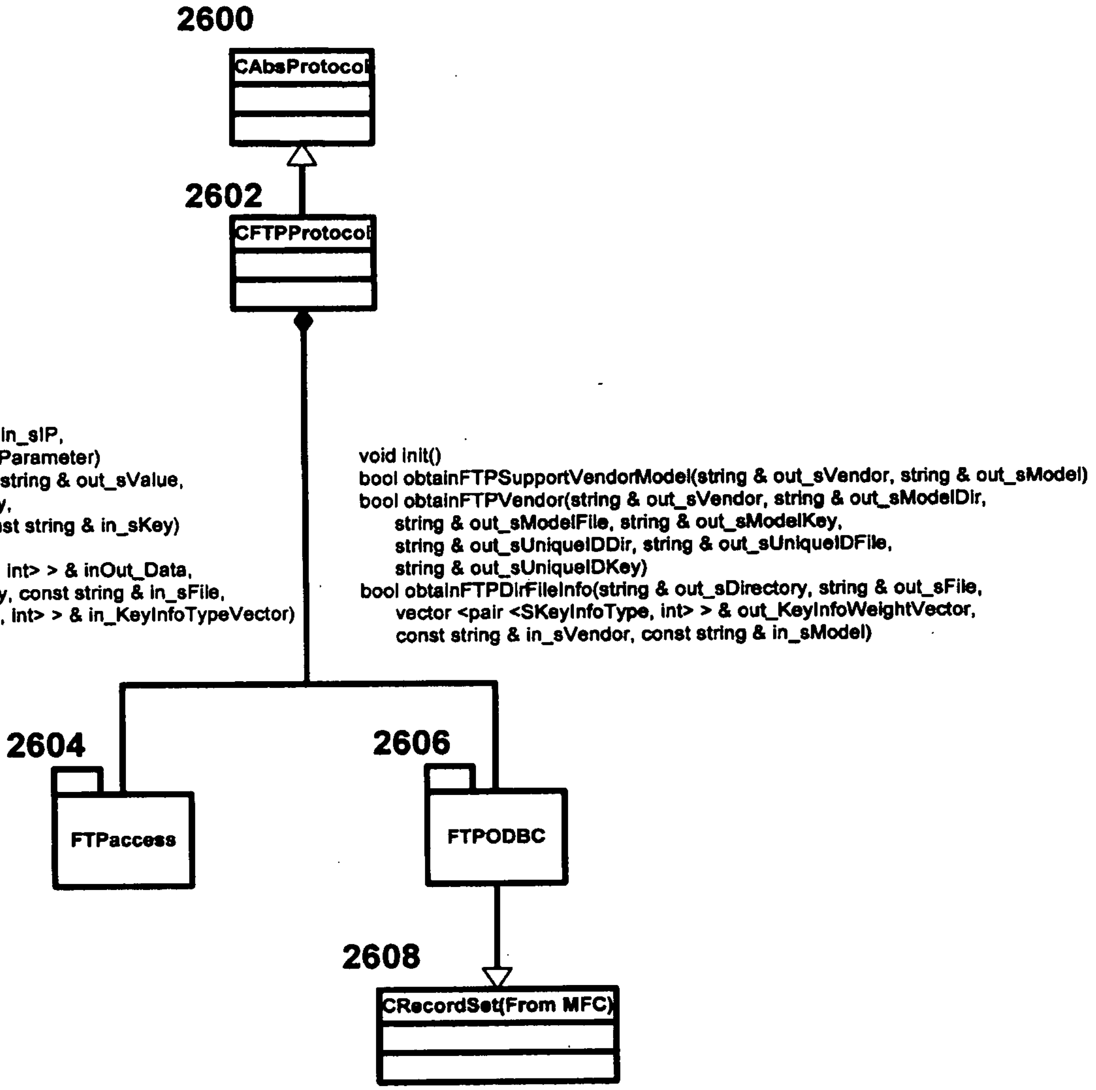
FIG. 24



```
bool initiateHTTP(const string & in_sIP,
     vector<SParameter> & in_Parameter,
     int in_nDelay=DefaultDelay)
bool obtainDataFromHTMLFile(
     map<infoType, pair<string, int> > & inOut_Data,
     const string & in_sWebPage,
     vector<pair<SKeyValueInfo, int> > & in_KeyValueInfoVector)
bool closeHTTP()
```

```
void init()
bool obtainHTTPSupportVendorModel(string & out_sVendor, string & out_sModel)
bool obtainHTTPVendor(string & out_sVendor, string & out_sWebPage,
     SKeyValueInfo & out_KeyValueInfo)
bool obtainHTTPDelay(int & out_nDelay, const string & in_sVendor,
     const string & in_sModel)
bool obtainHTTPUniqueID(string & out_sWebPage,
     SKeyValueInfo & out_KeyValueInfo, const string & in_sVendor)
bool obtainHTTPWebPageKeyValueInfo( string & out_sWebPage,
     vector<pair<SKeyValueInfo, int> > & out_KeyValueInfoWeightVector,
     const string & in_sVendor, const string & in_sModel)
```

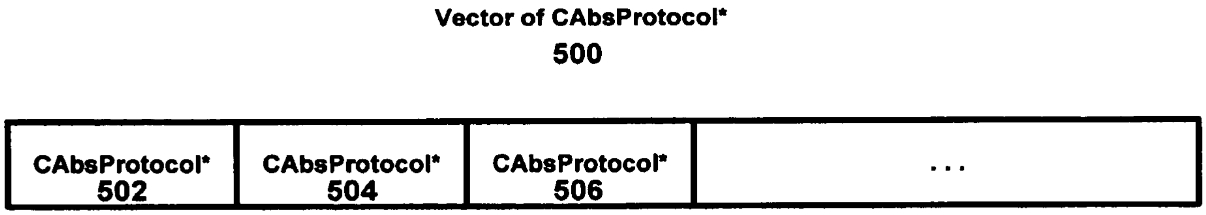
FIG. 25



```
bool initiateFTP(const string & in_sIP,
     vector<SParameter> & in_Parameter)
bool obtainValueFromFTPFile(string & out_sValue,
     const string & in_sDirectory,
     const string & in_sFile, const string & in_sKey)
bool obtainDataFromFTPFile(
     map<infoType, pair<string, int> > & inOut_Data,
     const string & in_sDirectory, const string & in_sFile,
     vector<pair<SKeyInfoType, int> > & in_KeyInfoTypeVector)
bool closeFTP()
```

```
void init()
bool obtainFTPSupportVendorModel(string & out_sVendor, string & out_sModel)
bool obtainFTPVendor(string & out_sVendor, string & out_sModelDir,
     string & out_sModelFile, string & out_sModelKey,
     string & out_sUniqueIDDir, string & out_sUniqueIDFile,
     string & out_sUniqueIDKey)
bool obtainFTPDirFileInfo(string & out_sDirectory, string & out_sFile,
     vector <pair <SKeyInfoType, int> > & out_KeyInfoWeightVector,
     const string & in_sVendor, const string & in_sModel)
```

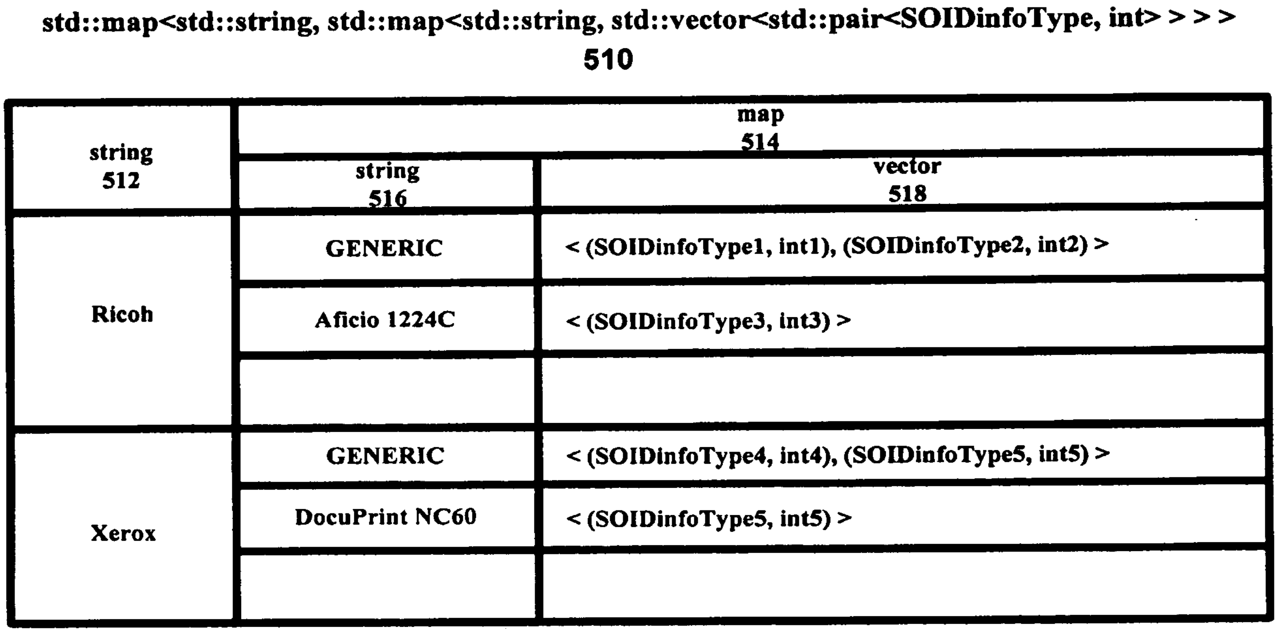
FIG. 26



Vector of CAbsProtocol*
500

CAbsProtocol* 502	CAbsProtocol* 504	CAbsProtocol* 506	. . .

FIG. 27A



std::map<std::string, std::map<std::string, std::vector<std::pair<SOIDinfoType, int> > > >
510

string 512	map 514	
	string 516	vector 518
Ricoh	GENERIC	< (SOIDinfoType1, int1), (SOIDinfoType2, int2) >
	Aficio 1224C	< (SOIDinfoType3, int3) >
Xerox	GENERIC	< (SOIDinfoType4, int4), (SOIDinfoType5, int5) >
	DocuPrint NC60	< (SOIDinfoType5, int5) >

FIG. 27B

std::map<std::string, std::map<std::string, std::vector<SWebPageInfo> > >
520

string 522	map 524	
	string 526	vector 528

Ricoh	Aficio X	Vector of struct SWebPageInfo {
		std::string m_sWebPage;
		std::vector<std::pair<SKeyValueInfo, int> >
		m_KeyValueInfoVector;
		};

Xerox	N4025	

FIG. 27C

std::map<std::string, std::map<std::string, std::vector<SDirFileStatusInfo> > >
530

string 532	map 534	
	string 536	vector 538

Ricoh	Aficio	vector of struct SDirFileStatusInfo {
		std::string m_sDirectory;
		std::string m_sFile;
		std::vector<std::pair<SKeyInfoType, int> >
		m_KeyInfoTypeVector;
		};

Xerox	N4025	

FIG. 27D

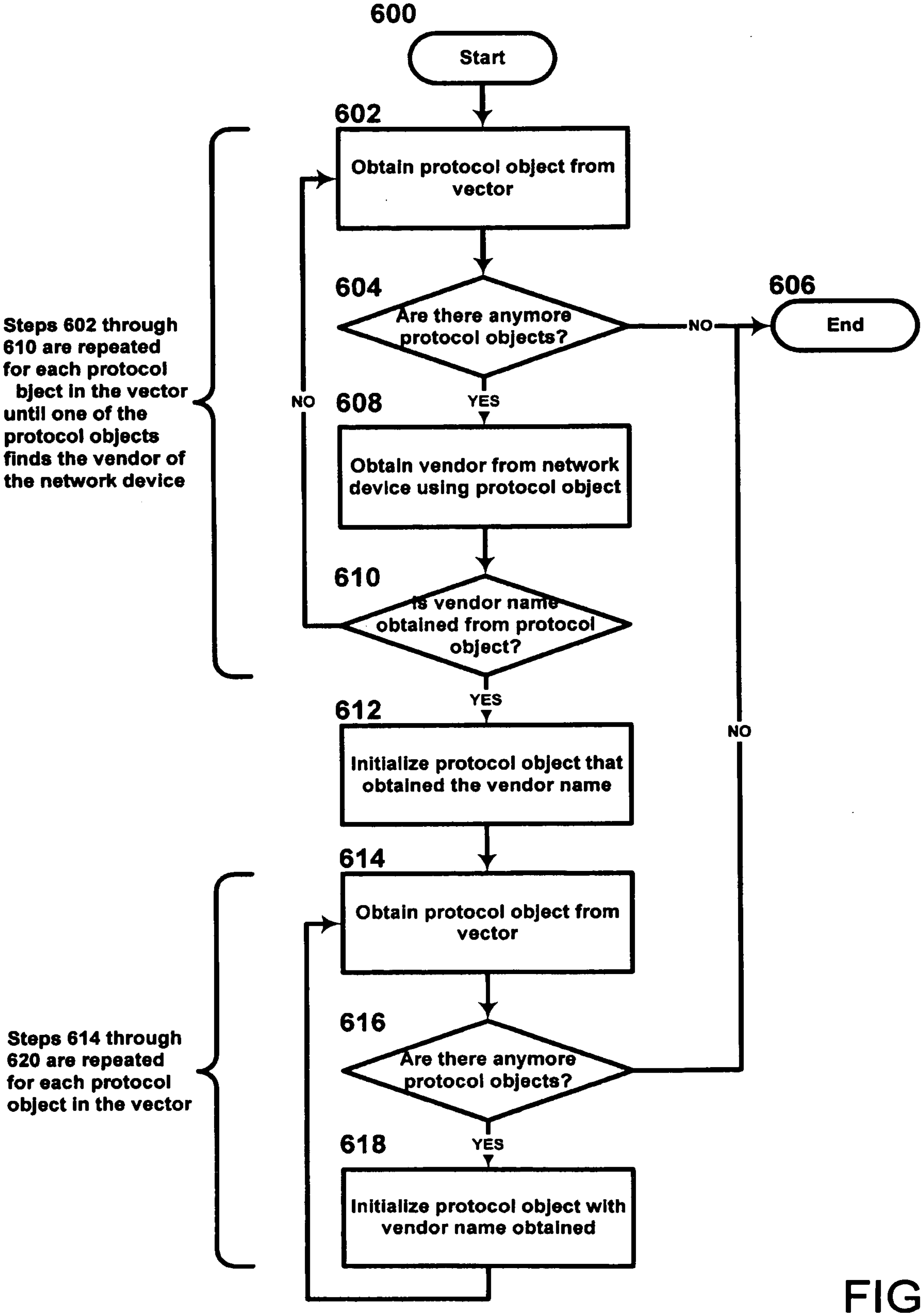
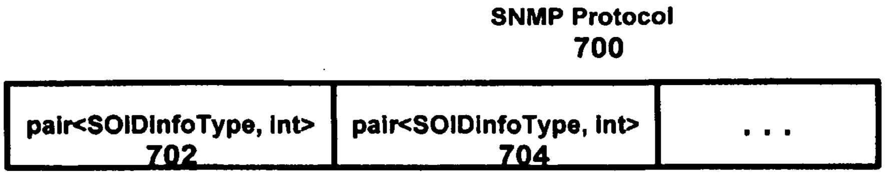

600
Start
602
Obtain protocol object from vector
604
Are there anymore protocol objects?
NO
606
End
YES
NO
608
Obtain vendor from network device using protocol object
610
Is vendor name obtained from protocol object?
YES
612
Initialize protocol object that obtained the vendor name
NO
614
Obtain protocol object from vector
616
Are there anymore protocol objects?
618
YES
Initialize protocol object with vendor name obtained
Steps 602 through 610 are repeated for each protocol bject in the vector until one of the protocol objects finds the vendor of the network device
Steps 614 through 620 are repeated for each protocol object in the vector

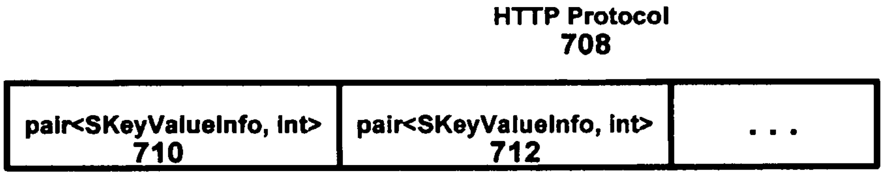
FIG. 28



706

```
struct SOIDInfoType {
   InfoType m_infoType;
   std::string m_sOID;
   SOIDInfoType();
   ~SOIDInfoType();
   void clear();
};
```

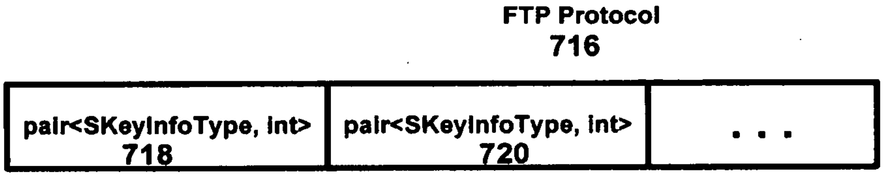
FIG. 29A



714

```
struct SKeyValueInfo {
   InfoType m_infoType;
   std::string m_sKey;
   int m_nPosition;
   std::string m_sType;
   std::string m_sDelimiter;
   int m_nInLinePosition;
   SKeyValueInfo();
   ~SKeyValueInfo();
   void clear();
};
```

FIG. 29B



722

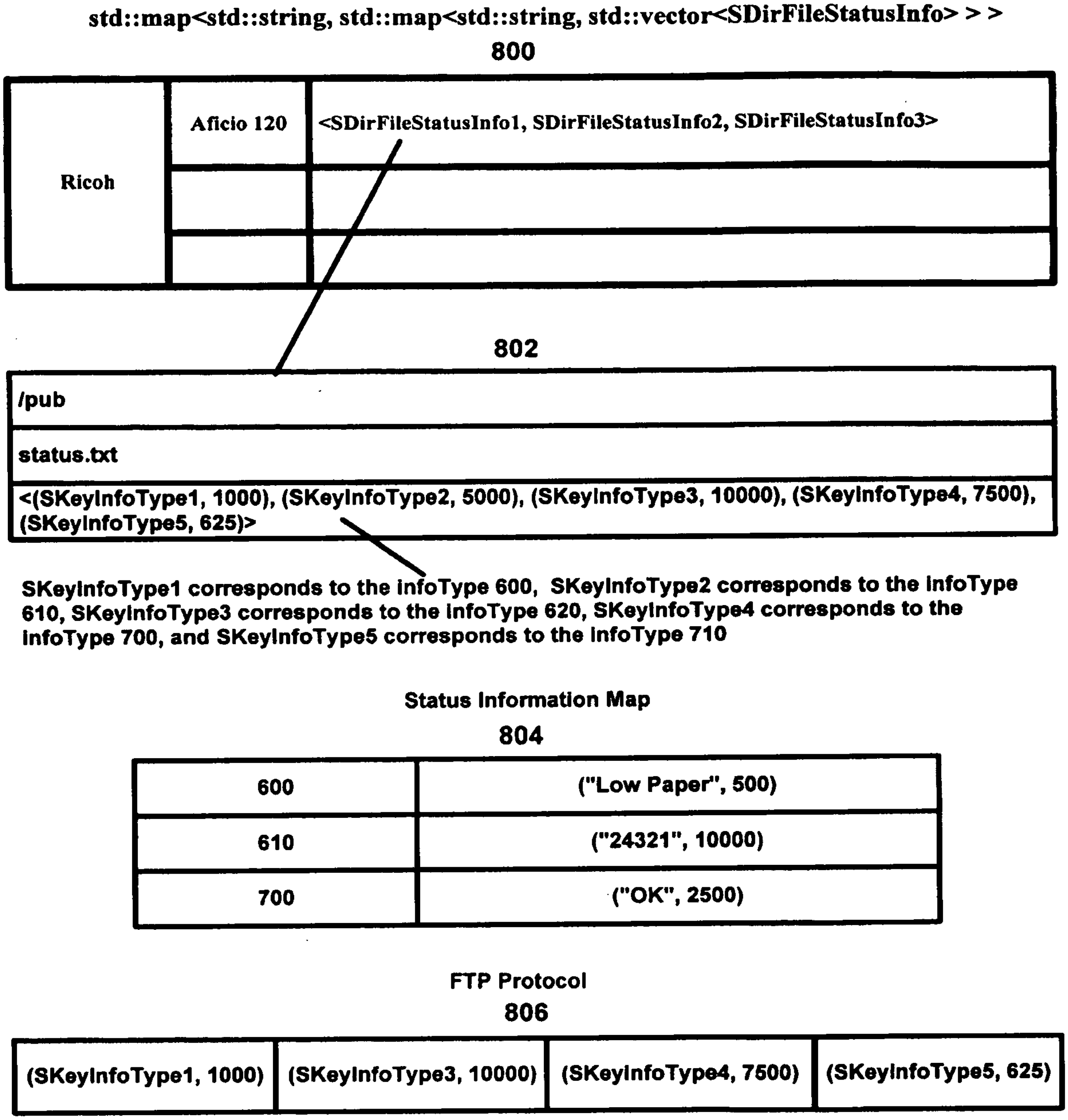
```
struct SKeyInfoType {
   InfoType m_infoType;
   std::string m_sKey;
   SKeyInfoType();
   ~SKeyInfoType();
   void clear();
};
```

FIG. 29C

Status Information Map
724

726	728
infoType	pair<string, int>
infoType	pair<string, int>
infoType	pair<string, int>
. . .	. . .

FIG. 29D



std::map<std::string, std::map<std::string, std::vector<SDirFileStatusInfo> > >

800

Ricoh	Aficio 120	<SDirFileStatusInfo1, SDirFileStatusInfo2, SDirFileStatusInfo3>

802

/pub
status.txt
<(SKeyInfoType1, 1000), (SKeyInfoType2, 5000), (SKeyInfoType3, 10000), (SKeyInfoType4, 7500), (SKeyInfoType5, 625)>

SKeyInfoType1 corresponds to the InfoType 600, SKeyInfoType2 corresponds to the InfoType 610, SKeyInfoType3 corresponds to the InfoType 620, SKeyInfoType4 corresponds to the InfoType 700, and SKeyInfoType5 corresponds to the InfoType 710

Status Information Map

804

600	("Low Paper", 500)
610	("24321", 10000)
700	("OK", 2500)

FTP Protocol

806

(SKeyInfoType1, 1000)	(SKeyInfoType3, 10000)	(SKeyInfoType4, 7500)	(SKeyInfoType5, 625)

FIG. 30

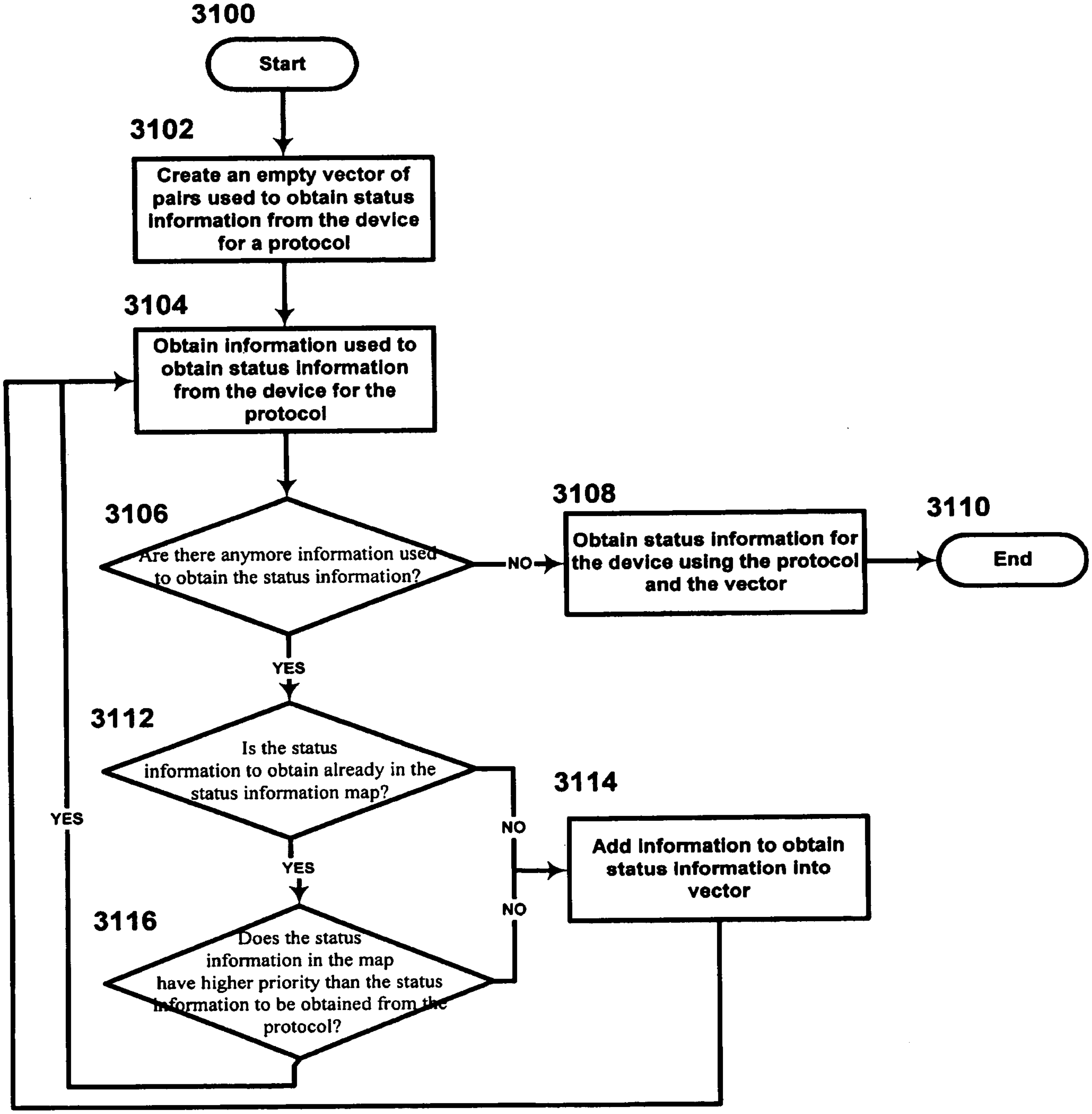

3100
Start
3102
Create an empty vector of pairs used to obtain status information from the device for a protocol
3104
Obtain information used to obtain status information from the device for the protocol
3106
Are there anymore information used to obtain the status information?
NO
3108
Obtain status information for the device using the protocol and the vector
3110
End
YES
3112
Is the status information to obtain already in the status information map?
NO
3114
Add information to obtain status information into vector
YES
NO
3116
Does the status information in the map have higher priority than the status information to be obtained from the protocol?
YES

FIG. 31A

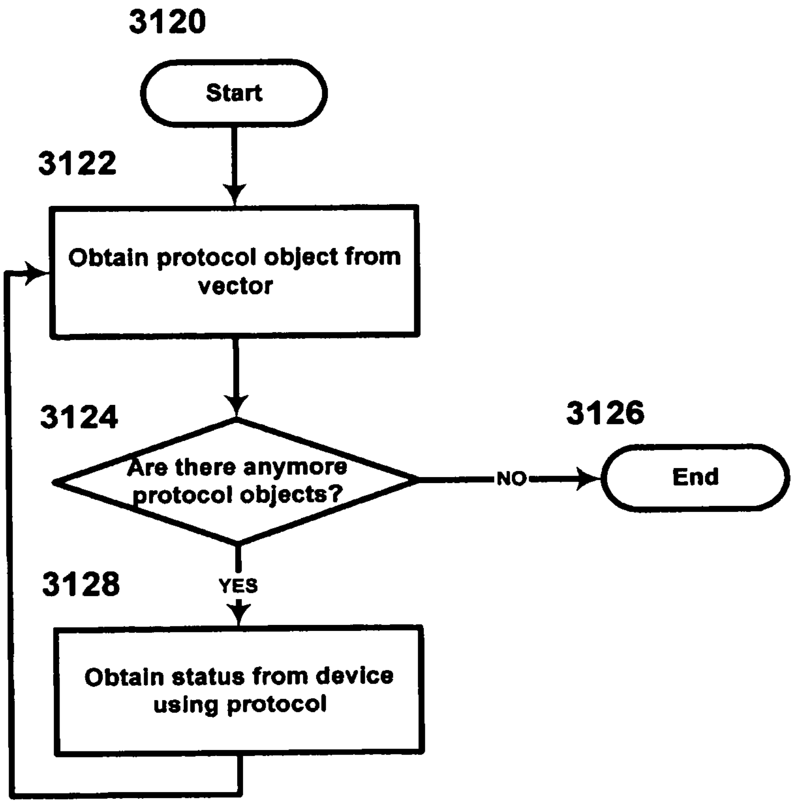

3120
Start
3122
Obtain protocol object from vector
3124
Are there anymore protocol objects?
NO
3126
End
YES
3128
Obtain status from device using protocol

FIG. 31B

Vendor Model Support Map
3200

3202	3204
string	int
string	int
string	int
string	int
. . .	. . .

FIG. 32A

Sample Vendor Model Support Map
3206

3208	3210
Xerox%%%%%NC60	1
Xerox%%%%%N4025	1
HP%%%%%LaserJet 9000	1
HP%%%%%LaserJet 4550	1
. . .	. . .

FIG. 32B

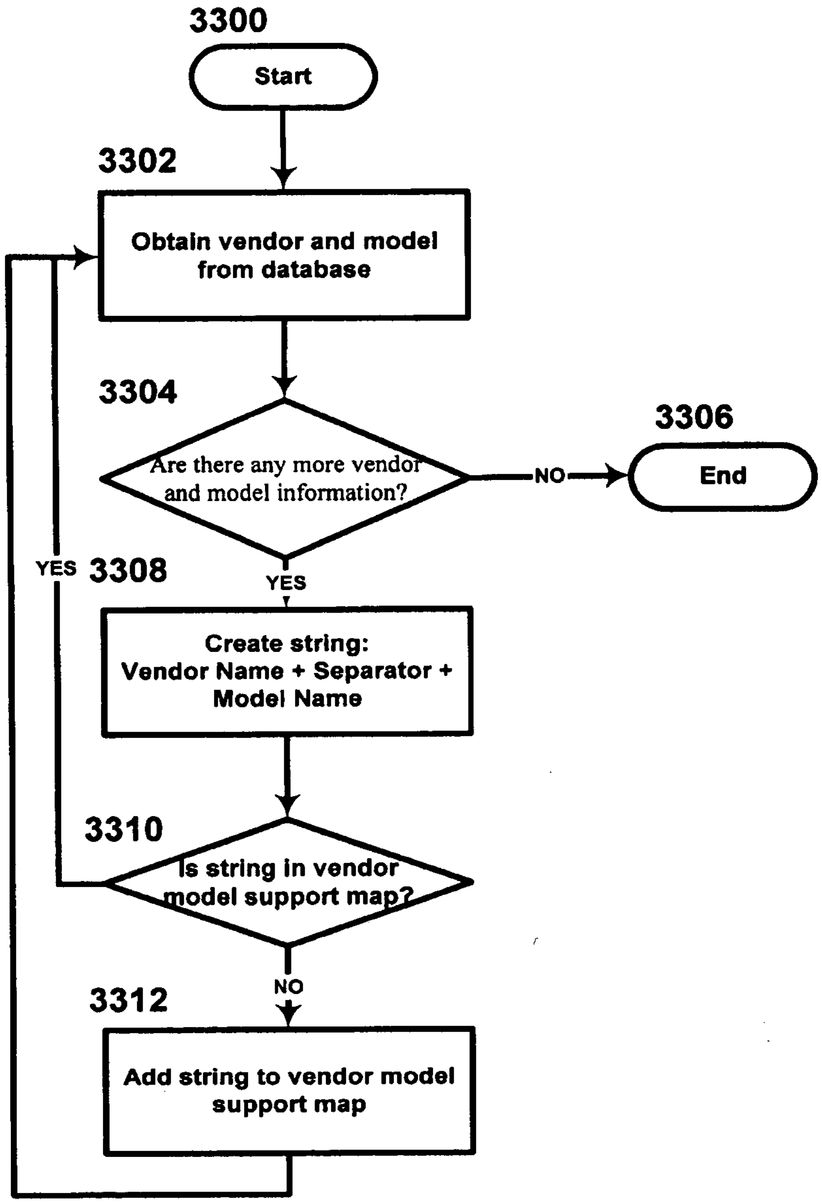

3300
Start
3302
Obtain vendor and model from database
3304
Are there any more vendor and model information?
NO
3306
End
YES
3308
Create string: Vendor Name + Separator + Model Name
3310
Is string in vendor model support map?
YES
NO
3312
Add string to vendor model support map

FIG. 33

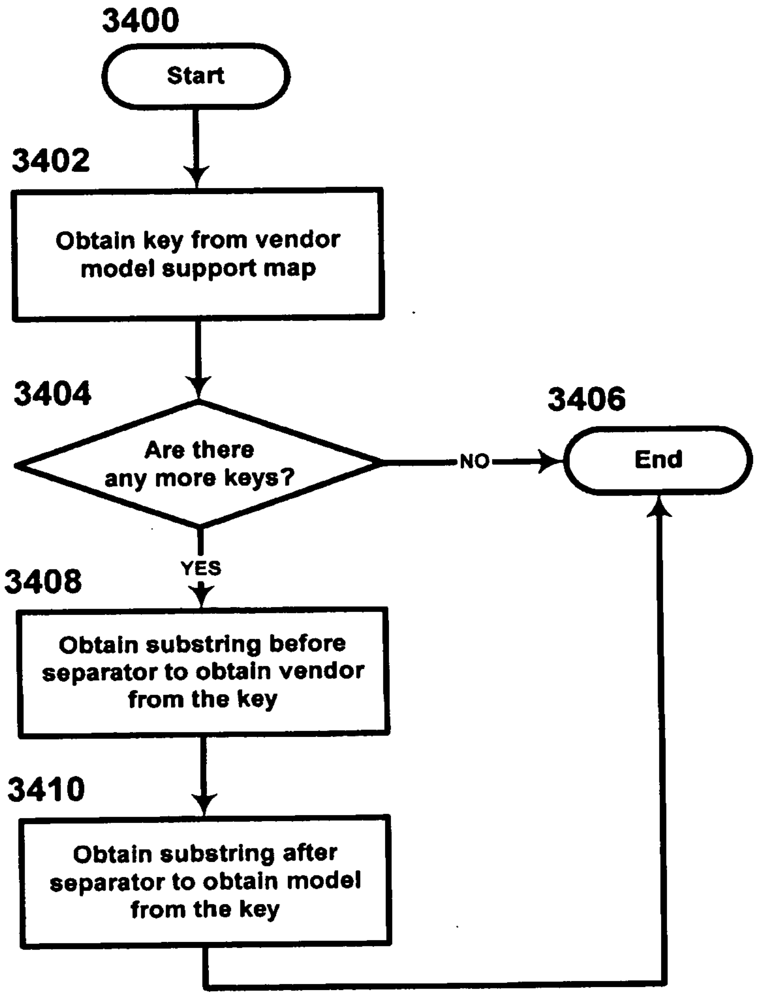

3400
Start
3402
Obtain key from vendor model support map
3404
Are there any more keys?
NO
3406
End
YES
3408
Obtain substring before separator to obtain vendor from the key
3410
Obtain substring after separator to obtain model from the key

FIG. 34

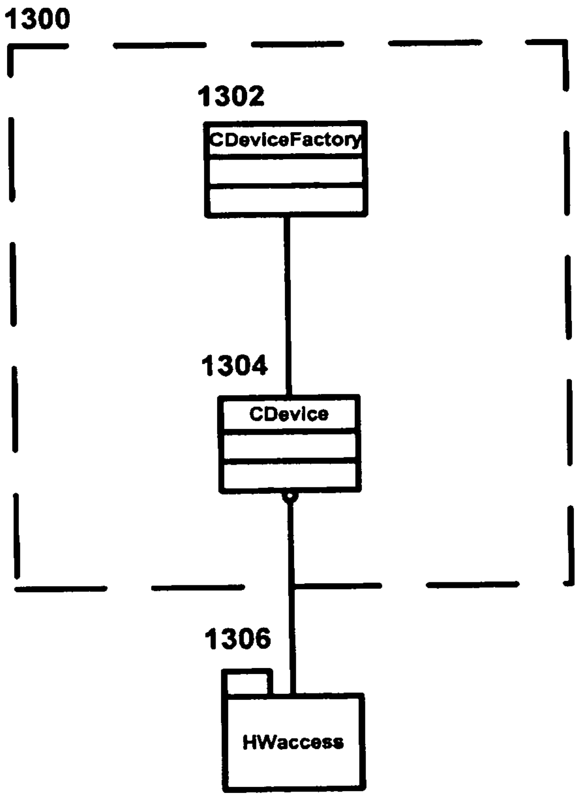


FIG. 35

Protocol Parameter Map
1400

1402	1404
string	vector<SParameter>
string	vector<SParameter>
string	vector<SParameter>
string	vector<SParameter>
. . .	. . .

1406

```
struct SParameter {
   std::string m_sParName;
   std::string m_sParValue;
   SParameter ();
   ~SParameter ();
   void clear();
};
```

FIG. 36A

1410

SNMP	<(COMMUNITY, private)>
FTP	<(USERNAME, abc), (PASSWORD, xyz)>

FIG. 36B

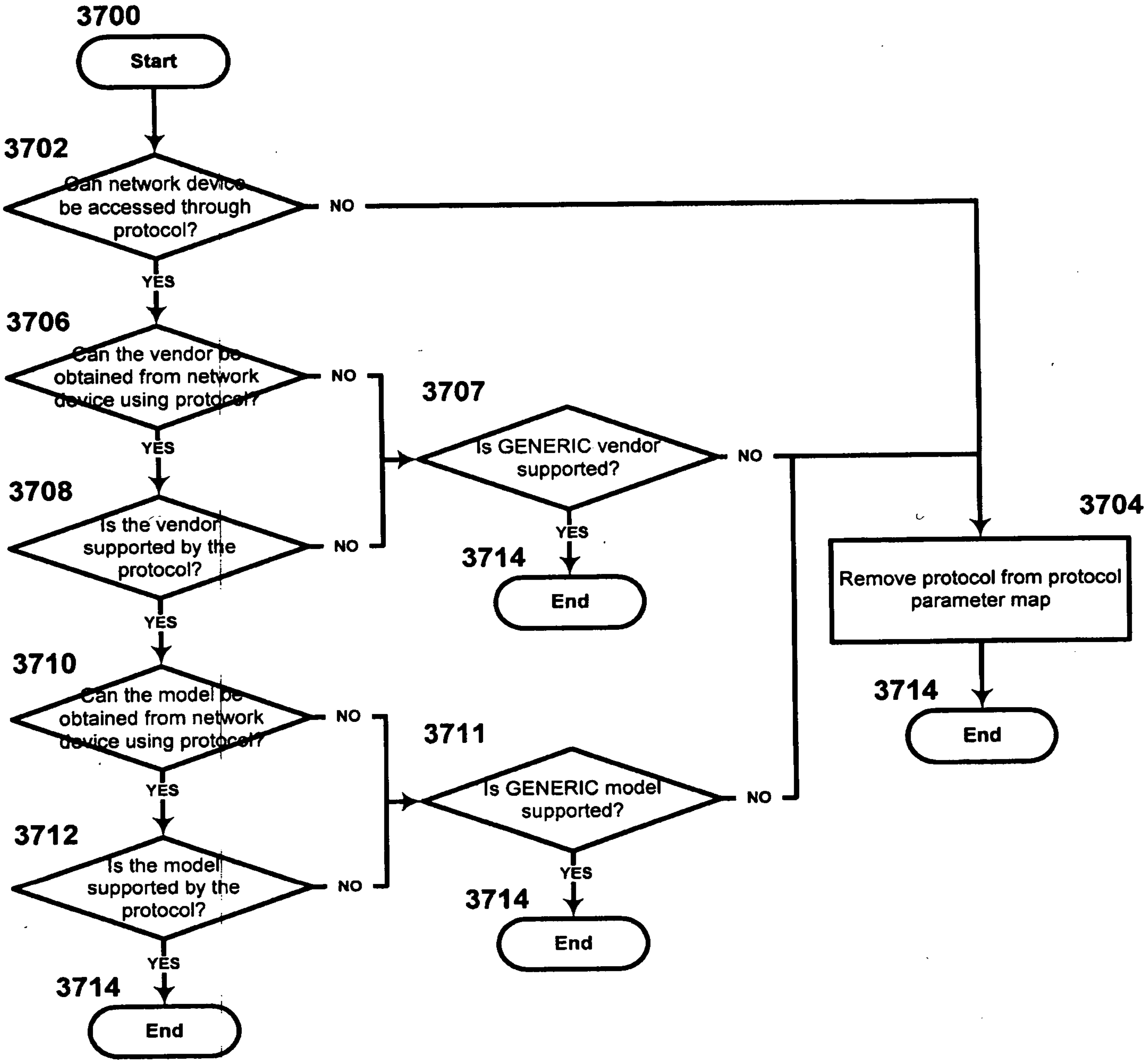

3700
Start
3702
Can network device be accessed through protocol?
NO
YES
3706
Can the vendor be obtained from network device using protocol?
NO
YES
3707
Is GENERIC vendor supported?
NO
YES
3714
End
3708
Is the vendor supported by the protocol?
NO
YES
3704
Remove protocol from protocol parameter map
3714
End
3710
Can the model be obtained from network device using protocol?
NO
YES
3711
Is GENERIC model supported?
NO
YES
3714
End
3712
Is the model supported by the protocol?
NO
YES
3714
End

FIG. 37